2

210600-30-T
Phase II Report
Draft - June 1990
Final - March 1991

A System for Mailpiece ZIP Code Assignment through Contextual Analysis

A.M. Gillies
M.P. Whalen
D. J. Hepp
J. M. Trenkle

Submitted to:
U.S. Postal Service
Office of Advanced Technology
Technology Resource Department
Washington, D.C. 20260-8121
Attn: Carl O'Connor

ERIM
P.O. Box 8618
Ann Arbor, MI 48107-8618

U S. Postal Service

REPORT ABSTRACT

Report Number [illegible]

1 Submitting U S Postal Service Organization

Technology Resource Department Office of Advanced Technology

2 Library Liaison Officer

Susan Schaeffer

3 Telephone Number

202-268-3867

4 Date Form Completed

2/28/91

5 Report Title

A Prototype System for the Contextual Analysis of Address Block Images

6 Report Date

June 1990

7 Organizational Author *(Requiring Group or Department, Ad Hoc Task Force, or Contractor, as applicable)*

Environmental Research Institute of Michigan

8 Contract Number 104230-86-H-0042 Task: 104230-88-D-2575

9 Restrictions

No Restrictions X **Official Use Only** __ **Limited Official Use Only** __

10 Contact *Personal Author, Project Manager, Project Director, etc.*

Andiew M. Gillies

11 Telephone Number

313-994-1200

12 **Descriptors**
(Subjects, Keywords, etc.)

Contextual Analysis	Segmentation
Address Block Interpretation	Automatic Feature Generation
Word Recognition	Feature Detection
Word Verification	Optical Character Recognition
Directory Matching	

13 Abstract of Report

This report describes the continued development and testing of a system for contextual analysis of machine printed address block images. The system receives a binary image of the address block (location of the address block is not a part of this work) and then: 1) segments the image into lines, words, and characters, 2) parses the address block to assign roles to the various words, 3) forms queries based on number of words in the street name, word lengths, identification of suffix and directional words, and (externally supplied) knowledge the ZIP code on the mailpiece, 4) retrieves street records from a postal directory, 5) matches the street names from the directory to the word images using a character confidence measure, 6) given a close match uses (externally supplied) knowledge of the street number to assign a 4 digit add-on code. It should be stressed that this system does not use isolated character recognition to read words, but rather a lexicon based word verification system. The report emphasizes description of the segmentation processes, an automatic feature generation method used in the word verification system, and the structure of the postal database. An authorized test, designed to allow comparison of results between contractors, was performed. Of the 929 address block images in the test, 348 (37.5%) received a correct 9-digit ZIP Code, 137 (14.8%) received a correct 5-digit ZIP Code, 170 (18.3%) received an incorrect 5 or 9-digit Code, and 272 (29.3%) were rejected by the system. After some small system refinements another test was run on independent images. Of the 959 images in this test, 430 (44.8%) received a correct 9-digit ZIP Code, 156 (16.3%) received a correct 5-digit ZIP Code, 97 (10.1%) received an incorrect 5 or 9-digit Code, and 276 (28.8%) were rejected by the system.

Table of Contents

1 Introduction

This report describes the activities in the Advanced Research in the use of Contextual Information for Automatic Postal Address Interpretation conducted for the United States Post Office (USPS) at the Environmental Research Institute of Michigan (ERIM) under contract 104230-86-H-0042, Task No. 104230-88-D-2575. During Phase I of this project, an initial prototype Contextual Analysis (CA) system was constructed. The Phase II activities concentrated on refining the individual components of this system. Near the end of Phase II, an authorized test designed to allow comparison of results between contractors was performed. Of the 929 address block images in the test, 348 (37.5%) received a correct 9-digit ZIP Code, 137 (14.8%) received a correct 5-digit ZIP Code, 170 (18.3%) received an incorrect 5 or 9-digit Code, and 272 (29.3%) were rejected by the system. After some small system refinements another test was run on independent images. Of the 959 images in this test, 430 (44.8%) received a correct 9-digit ZIP Code, 156 (16.3%) received a correct 5-digit ZIP Code, 97 (10.1%) received an incorrect 5 or 9-digit Code, and 276 (28.8%) were rejected by the system.

This report describes the various components of the overall system, evaluates its performance, and identifies areas of possible improvement. Specifically, this report recommends improvements to the address block parser, the implementation of alternate character segmentation hypotheses for word images, an investigation into possible image repair techniques and their proper place for integration into the system. More importantly, Phase III of the project will focus on the integration of MLOCR information into the end-to-end system. The effects of this information on the structure and operation of such system components as word segmentation, character segmentation, address block parsing, strong database queries, and word verification, will be a central question in the Phase III effort.

1.1 Motivation

The United States Postal Service (USPS) deploys large numbers of optical character recognition (OCR) machines for letter sorting. Their goal is to sort every piece of mail to its appropriate ZIP+4 (9 digit) code. The majority of mailpieces, however, do not display the full 9 digits. This means that the address reader must be able to read the street address or PO Box number from the mailpiece in addition to the ZIP code or city and state. The street address must then be matched against the USPS postal directories to obtain the correct ZIP+4 code.

Letter sorting is made difficult by the variability in the fonts and print quality found on address block images. The small amount of text in each address makes it difficult for the system to adapt to the characteristics of each mailpiece. Omissions and errors in the address, so called patron errors, further complicate the problem. The information redundancy in addresses allows some of these errors to be overcome.

Current readers based on isolated character recognition are limited by their character recognition rate. Extensive research over the past several decades [1,2,3,4,5] has been unable to increase this rate to the level needed to support letter sorting goals. For this reason we wish to investigate methods for the contextual analysis of address block images.

A contextual analysis system uses information about words it is likely to encounter and the arrangements in which they are most commonly found to arrive at the best global interpretation of the address block image. This means that the system must be able to bring to bear many disparate sources of information, including postal directories, addressing conventions, and feature evidence from the image itself, in analyzing an address block. In doing so, the system must be able to entertain multiple hypotheses at different levels in an information hierarchy during the analysis process.

This report describes a system that uses a combination of context-based vision, mathematical morphology, model-based vision, and reasoning with uncertainty to move towards a solution to this difficult problem. It features a top-down, two-staged control strategy that employs a hypotheses generation and test paradigm. Within this top-down system, mathematical morphology is used to extract size and shape information; model-based vision is used to parse address blocks and create hypotheses structures; and uncertain reasoning techniques are used to evaluate these hypotheses.

1.2 Project Goals

The objective of this research program is to investigate the potential use of contextual information at various image processing and understanding levels. This ongoing work is expected to lead to an eventual improvement in OCR recognition performance, and thus result in significant reduction in the amount of labor required to sort mail. The system designed to achieve this goal will overcome some of the limitations of previous OCR systems. In particular, the system described in this document will resolve many of the problems attributed to poor quality writing, multiple fonts and scripts, and segmentation errors that are characteristic of current readers.

1.3 Project Datasets

The data used for this project were binary address block images. They were supplied by Electrocom Automation (ECA) and were acquired using a current OCR machine with image capture facilities processing mailpieces from a live mailstream. In this respect, the data was identical to what would be expected if the developmental system was integrated into the USPS remote video encoding (RVE) system.

Associated with each binary image is a truth record recording the actual contents of the image as read by a human. For each word in the image, this information consisted of the image name, the image coordinates of the bounding box of the word, the string corresponding to the word, the meaning code of the word, and the line number that the word is on. The meaning code is a two letter designation of the role played by a word in the image. Figure 1.3-1 shows some examples of meaning codes and words that could fill their given role.

Meaning Code	Definition	Examples
CW	City Word	Dallas, Ann Arbor
SW	State Word	Texas, Michigan
PP	Postal Phrase	CAR-RT, SORT

Figure 1.3-1 Meaning Codes

In addition to the truth record, ECA supplied a level of sortation (LOS) record for each address block image. The LOS record gives the ZIP code assigned by the MLOCR machine, the ZIP code present on the mailpiece (if any) as read by a human, and a list of valid ZIP codes for all possible levels of sortation. This latter information was retrieved through human interaction with the NDSS lookup software. As an example, consider a business firm located in a suite in a high rise office building. One possible 9 digit ZIP Code could be for the block face of the building. Another ZIP Code could be for the particular building that the firm is located in. A more specific ZIP Code might exist for the floor that the firm is on. The firm might even have a 9 digit code specifically for itself. The LOS information is used to grade contextual analysis system's responses with regard to both correctness (is the response in the list of valid responses) and fineness of sort.

The collected data were broken down into 12 categories based on two parameters: mailpiece type, and OCR sortation level. There are four possible values for mailpiece type, each of which is listed below along with its definition:

B	Bulk Business Mail
C	Collection Mail
D	Destinating Mail
M	Metered Mail

There are three possible classes of OCR sortation level. A sortation level class of 5 means that the current MLOCR machine attached a 5-digit ZIP Code to the mailpiece. Similarly, a class 9 mailpiece received a 9-digit ZIP Code and a class R mailpiece was rejected by the current machine.

Using this information in an example of a mailpiece category, C5M corresponds to all of the collection mailpieces that had a 5-digit ZIP Code attached to them. There are also different print types processed by the OCR machine (e.g. machine, hand printed, etc.). The M in C5M refers to the fact that all of the images in this category are machine printed and, in fact, all of the images used for this project were machine printed.

The data was broken up into a training dataset and a testing dataset taking images rrom each ot tne twelve categories. The testing dataset was further broken down into three subsets to provide for incremental testing of the system. Figure 1.3-2 summarizes the number of images in each dataset broken down into the 12 previously mentioned categories.

Mail Class	Dataset				
	Training 1	Test 1.1	Test 1.2	Test 1.3	Test Set 1 (1.1 + 1.2 + 1.3)
C5M	550	151	151	148	450
C9M	110	34	34	32	100
CRM	440	134	134	132	400
M5M	550	151	151	148	450
M9M	110	34	34	32	100
MRM	440	134	134	132	400
D5M	450	151	151	148	450
D9M	90	31	31	28	90
DRM	360	121	121	118	360
B5M	200	59	59	57	175
B9M	40	12	12	11	35
BRM	200	41	41	38	120
Dataset Totals	3540	1053	1053	1024	3130

Figure 1.3-2 Dataset Analysis

During Phase I of the project, the system components were individually constructed and tested on images from the Training 1 dataset. During Phase II of the project, the individual system components were refined and trained on images from both the Training 1 and Test 1.1 datasets. At the end of phase II a test using images from test set 1.2 was performed. The results of this test will be described later in this report.

1.4 Project Outcomes

This research and development project has produced several major outcomes. Included within these outcomes are the development of sophisticated preprocessing techniques that reliably correct tilt present in the raw imagery, the development of techniques to detect and segment line images from a parent address block image, and detect and segment word images from these lines; the development of a highly successful word verification subsystem; the development of a model-based parsing component that propagates multiple parse hypotheses; the development of an advanced query generation engine; the development of a knowledge representation subsystem that uses an inverted database to efficiently store and retrieve information about addresses; and the development and refinement of an end-to-end contextual analysis system that incorporates all of these techniques to process input address block images.

1.5 Future Directions

Although this research has been very successful, there is still much work that needs to be done. The primary objective of phase III of this project is the integration of contextual analysis with the OCR capabilities of current MLOCR machines. This may be accomplished by integrating MLOCR character recognition output several modules of the current system including: numeral reading, number vs. non-number recognition, word verification, and query formation.

In addition, the modules of the current system need refinement in several areas. The word and character segmentation modules should be changed to generate multiple hypotheses. Image processing methods should be augmented to include better image quality estimation, image repair capabilities, and punctuation removal. Numeral reading capabilities, apart from those available in the MLOCR, should be added to the system. The character normalization and feature extraction capabilities in the word verification module should be refined. The address block parsing module should be modified to include address block format statistics, and the address block models should be augmented to include more address formats. The query and database modules should be extended to handle new query types (for example number of digits in the street number), thesaurus replacement capabilities, and the NCWS database. The final decision module should be extended to better integrate the current MLOCR output with the contextual analysis results, and to provide address-level verification methods.

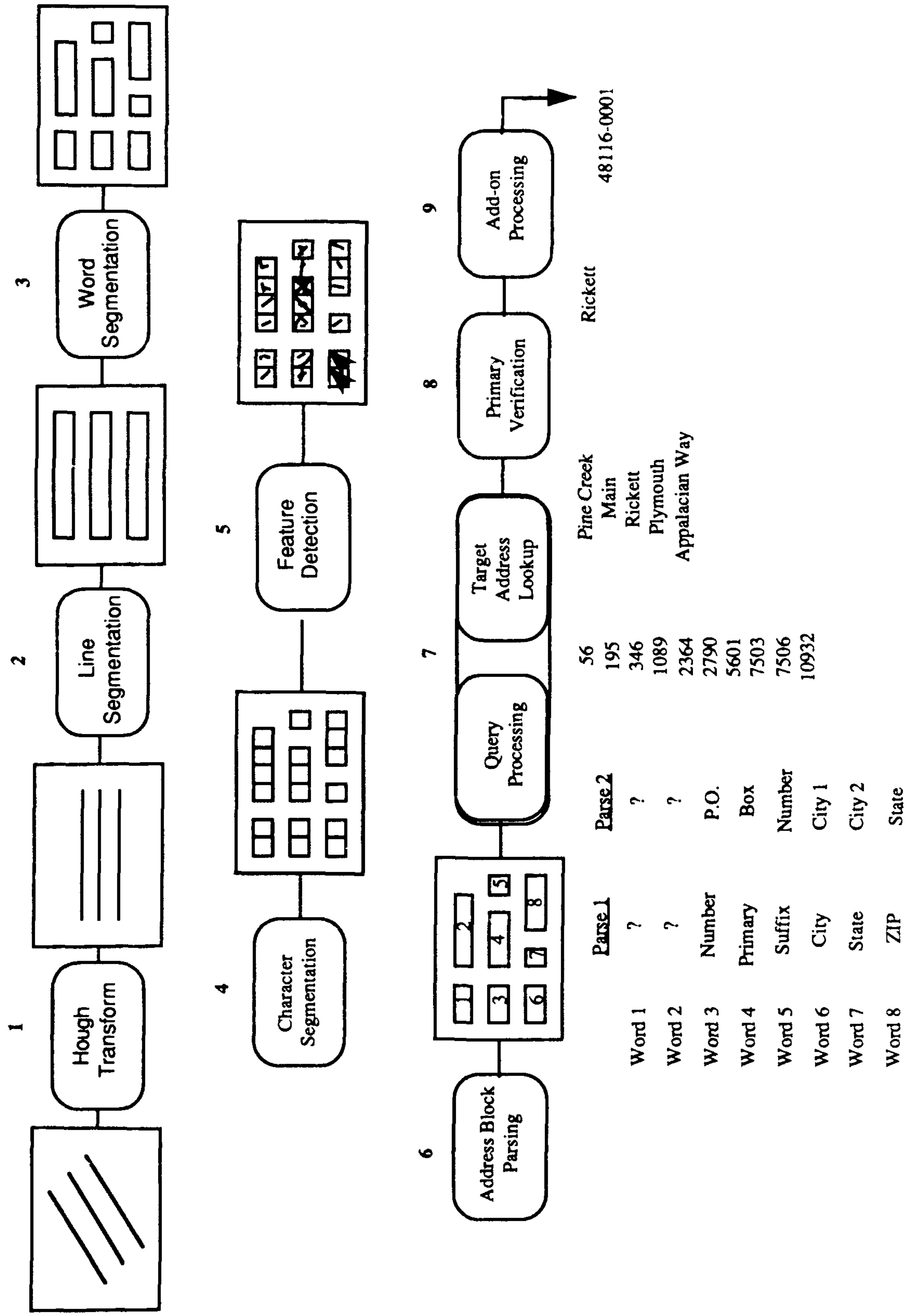
A complete plan for the next phase of this work (phase III) is given in section 4 of this report.

2 System Description

This section describes the architecture and function of the contextual analysis system as it stands at the end of phase II. The section starts with a system overview. Then, in a series of subsections, the major system modules are described. Each subsection details the functions performed by the module, and gives an analysis of the module's performance. In addition, important issues encountered in the development of the module and possible future refinements are discussed.

2.1 System Overview

An overview of the Contextual Analysis system is shown in Figure 2.1-1. Processing within this system is composed of nine distinct phases as seen in the figure. An input binary image is first handled by a Hough transforming routine to remove any tilt that might be present. A series of segmentation modules follows this tilt removal. Line segmentation isolates the individual lines of text within the address block image. Word segmentation isolates the words within each line. Character segmentation detects the boundaries between individual characters inside each word image. Once the necessary segmentation procedures have completed, a bank of morphological feature detectors is executed on the words of the address block image. The next phase of processing is address block parsing which uses the feature detection results in addition to the number and spatial arrangement of words to arrive at a set of likely parses for the image. The query processing module accepts these multiple parses and generates appropriate address block attributes from which to query an attribute database. These queries return target address numbers whose corresponding address records possess the required address attribute. These numbers are intersected and sent to a target address lookup procedure that retrieves the actual address records from their target numbers. The primary names from these records are extracted and sent to a primary verification module. This module chooses the most likely primary name from the set. A final add-on processing module combines this primary name with information previously extracted from the address block and target record to attach the correct ZIP code to the mailpiece.

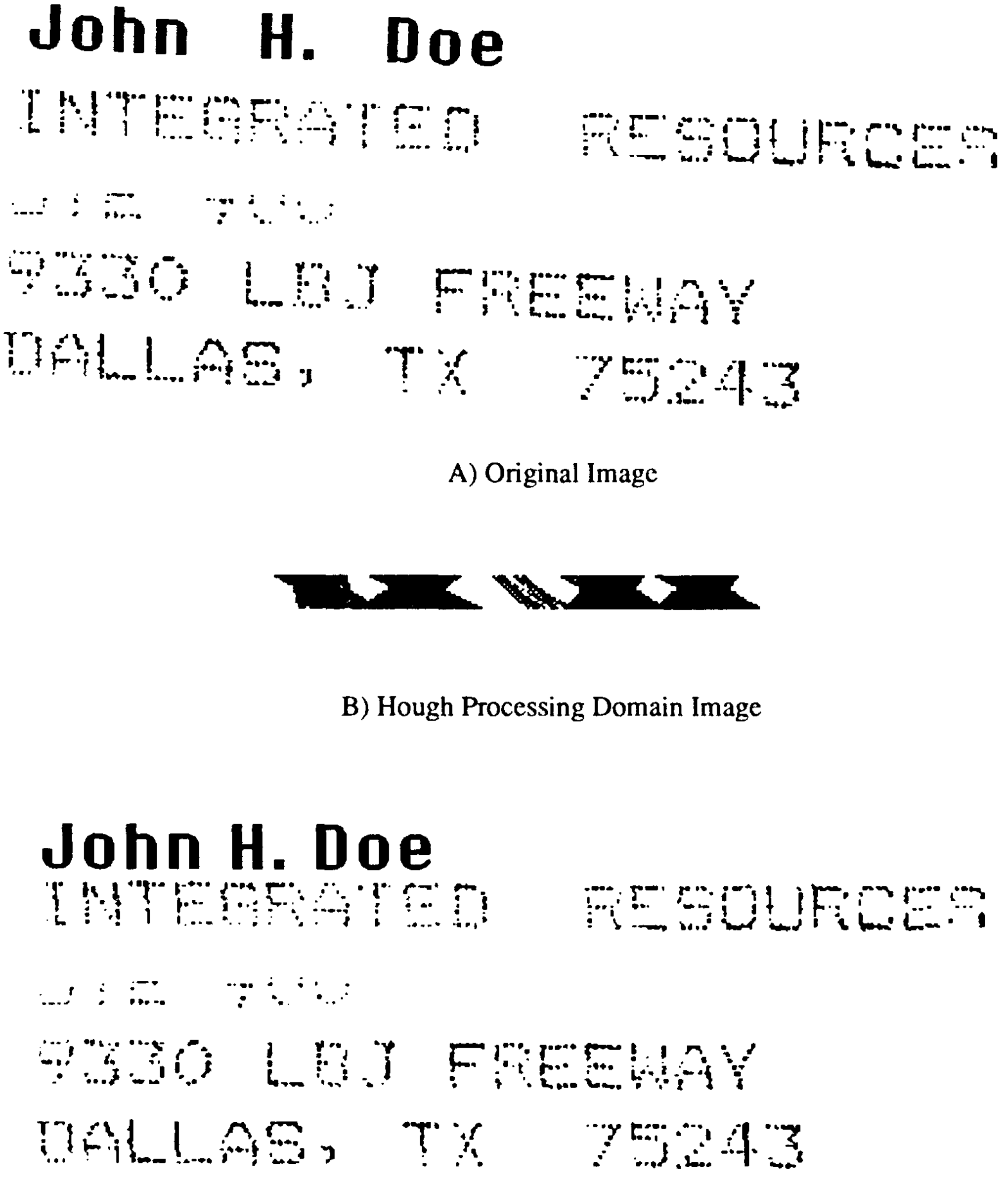


	Parse 1	Parse 2
Word 1	?	?
Word 2	?	?
Word 3	Number	P.O.
Word 4	Primary	Box
Word 5	Suffix	Number
Word 6	City	City 1
Word 7	State	City 2
Word 8	ZIP	State

Figure 2.1-1 End-to-end System

2.2 Hough Transform

Before processing of the address block images can begin, any tilt or skewing of the image must be corrected. Because the subsequent processing steps depend upon an input image with no tilt, it is imperative that as many of the readable images are successfully de-skewed as possible. Based on dead-letter mail, this problem occurs approximately 26% in rejected mail, 14% in default mail (5 or 9-digit code already present), 4% in 5-digit mail, and 2% in 9-digit mail (statistics from ECA).



A) Original Image

B) Hough Processing Domain Image

C) Corrected, Resampled Image

Figure 2.2-1 Hough Transform Example

This module uses a Hough Transform to determine the ang'e at which the address olock is tilted. The image is then resampled to produce an untilted image.

The detection of tilt angle via Hough transform is conducted in two stages. First, a coarse search is performed on the image to determine the tilt angle within a 10 degree range. A fine resolution search is then performed within this coarse range using 1/2 degree increments. Every fourth pixel is used in the horizontal direction to reduce the computation time. The optimal angle is determined using a contrast measure calculated from the local max and min of a given neighborhood in the Hough space. Figure 2.2-1 shows results from processing a sample image.

The deskewing algorithm correctly removes tilt in approximately 96% of the address block images. Its failures often occur on images for which further processing is not likely to arrive at a correct ZIP+4 code, and so do not contribute to system failures. This performance is judged to be acceptable at this time. The Hough transform algorithm is, however, computationally expensive. Future investigations of the deskewing process may focus on methods which are less computationally expensive.

2.3 Line Segmentation

The line segmentation algorithms were greatly enhanced during Phase II with a considerable improvement in performance. At testing for Phase I the line segmentation module was properly segmenting approximately 90% of the lines in address block images. After Phase II augmentation the module was performing at almost 97%.

Figure 2.3.1 shows a flowchart describing the line segmentation module. The important steps in the algorithm are numbered along the left margin and will be discussed in greater detail. Briefly, the module takes an address block image as input, verifies that this is a suitable image, applies segmentation algorithms to the extent necessary and returns the properly segmented line images.

Figure 2.3.2 shows several problems that cropped up during algorithm development. These problems were the primary motivators for developing specialized filters in the line segmentation module to deal with problems such as non-informative text lines, company logos and "Pay to the Order of" type blocks.

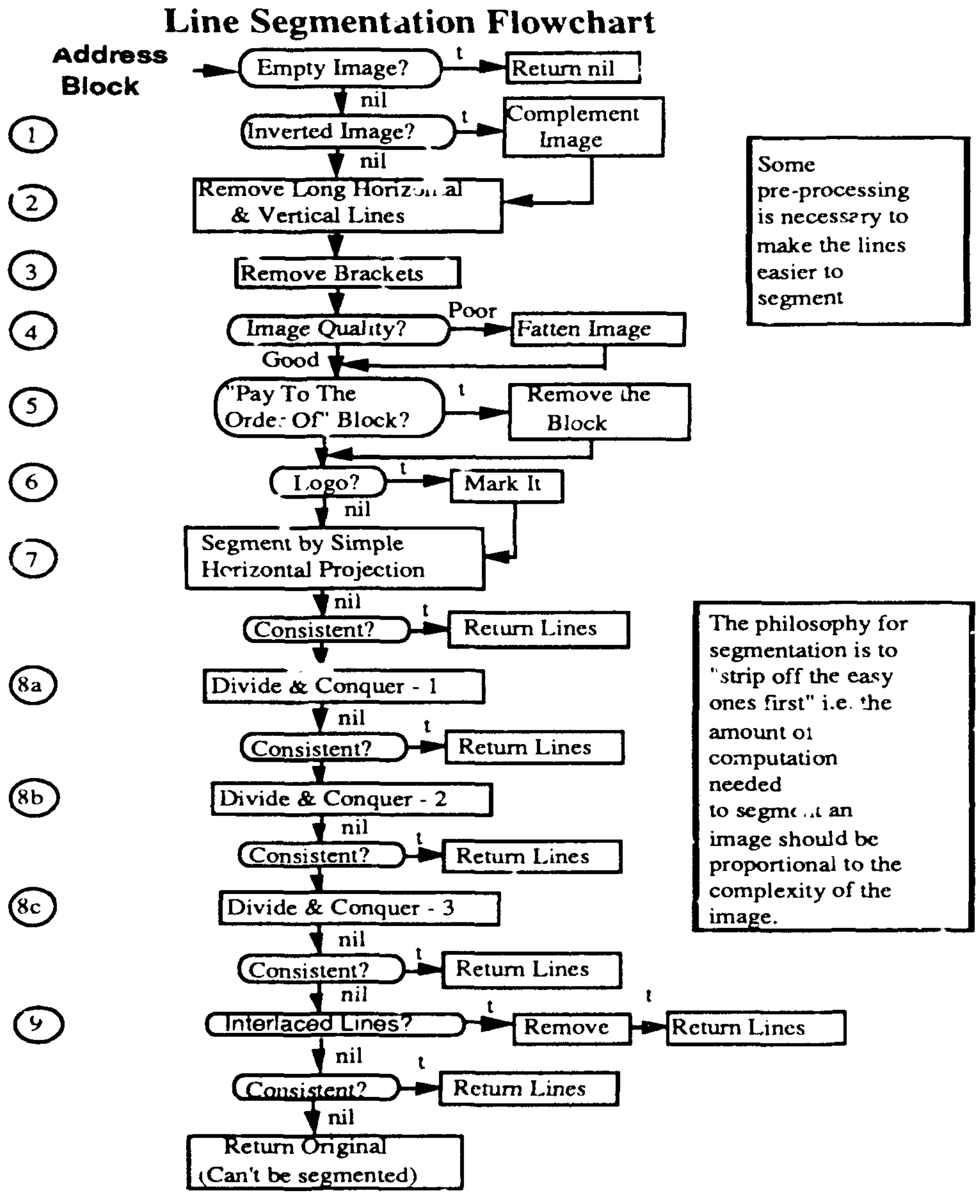
It should be emphasized that the algorithms for removal of noise, lines, etc. were designed with a conservative philosophy. The algorithms preserve potentially meaningful text at the expense of not removing some junk. The quality of the image is taken into account in these decisions.

Following is a more detailed description of the numbered steps in the line segmentation flowchart.

1) In some rare instances address blocks were inverted in the sense that the roles of background and foreground were reversed. In this form our algorithms could not properly segment the address block image because of our assumption that set pixels are text. A very inexpensive check for this situation was inserted in the system to allow us to detect and correct occurrences of this problem. The check is simply to calculate the ratio of set pixels to total pixels and if this value exceeds the empirically determined threshold 0.5 then the complement of the image is taken and used for further processing.

Figure 2.3.3 gives an example of this situation.

2) Another problem that may potentially interfere with segmentation is the presence of long horizontal and vertical lines in an address block image. In some of these address blocks the address is bordered on most or all sides by long connected lines, while in others, the address was printed on an envelope that had pre-printed lines. Vertical lines lead to problems in line segmentation because they cause the horizontal projection to be completely connected. Horizontal lines can also cause a variety of errors. Their remnants can be mistaken for words or parts of words in later segmentation. When segmenting lines into words they can cause the vertical projection to be completely connected. A morphological filter has been placed in the processing flow that removes these long lines from all address blocks. Figure 2.3.4 gives several examples of line removal processing.



Some pre-processing is necessary to make the lines easier to segment

The philosophy for segmentation is to "strip off the easy ones first" i.e. the amount of computation needed to segment an image should be proportional to the complexity of the image.

Figure 2.3.1

3) A minor problem, but one that can be easily corrected is the instances of address block images that have the address framed by large brackets located at the four corners of the block. These can have repercussions in a) line segmentation where the lines containing brackets may be merged with other nearby lines or the lines may be made too wide or b) word segmentation where the brackets may be taken as words or parts of other words. Figure 2.3.5 illustrates this problem.

4) A process that aids segmentation greatly is the attempt to rate the quality of an address block image based on the dot factor which is a morphological measure of the dottiness of an image (it is the ratio of the set pixels in the original image over the number of set pixels in the same image processed by some closing operations) - the higher the measure the worse the quality of the image. Also an attempt is made at the removal of noise caused by a number of things including the sensor on MLOCR being dirty and cellophane "windows" on business envelopes that produced a glare

when they were being digitized. Most of the noise observed in our data sets was spurious point noise that can be removed using some point filtering operations. Figure 2.3.6 presents several examples.

5) A situation that causes many problems in segmentation as well as parsing is the existence of non-informative text blocks on the left hand side of some address block images. These blocks come in several flavors:

"Pay to the Order of", "To:", "Attn:", "Bill To:", etc. They are affectionately known as PTOOS. These blocks are removed for two very important reasons. First, the text in these blocks is typically not aligned with the text in the address, so the horizontal projection does not clearly show line division. Secondly, even if the true line could be detected in the horizontal projection domain, its height was usually biased due to the non-informative text block. This is a very severe problem since our word verification features are parameterized on height. Figure 2.3.7 shows several examples of the PTOO problem and also points out the characteristics of PTOOS that allow them to be removed.

6) Company logos occur rather frequently and have the potential to cause problems in segmentation and parsing. Our line segmentation algorithms measure the height of each candidate line relative to the heights of other candidates. Most logos exceed the acceptable threshold and previously, were split up into many incorrectly segmented lines. Fortunately, we have found certain characteristics of company logos that allow them to be easily detected. For example, logos tend to have pictures or insignias that are much bolder than regular text. The text used in logos is also typically larger and bolder than normal text. These facts along with the fact that logos are usually the first objects in an address block allows us to morphologically detect a logo and avoid splitting it into troublesome lines. A variation of logos has also been seen in which the big insignia is placed on the left side of the address block. This situation is very similar to the PTOO problem and is dealt with in almost exactly the same way. Figure 2.3.8 gives examples of the two flavors of logos.

7) This step is the first and simplest attempt to segment the pre-processed address block image into lines. In this step the horizontal projection (a.k.a. horizontal pixel migration, horizontal histogram) of the address block image is taken and ideally each line in this projection will appear as a contiguous set of positive values whereas the absence of lines will be noted by values of zero in the projection. After detection of potential lines the variance is calculated. If the variance is below the empirically determined threshold then the line images so generated are used as the output of the line segmentation process. If the variance criterion is not satisfied the address block image is passed on to the next module for further segmentation processing. Figure 2.3.9 illustrates this process.

8) The more complex processing that constitutes the second level of line segmentation is referred to as the Divide & Conquer approach. It is an accurate name in the sense that the three possible steps in this level operate on only parts of the original address block image in their attempts to obtain a proper segmentation.

8a) The first step in Divide & Conquer processing is based on the realization that when lines are touching they usually have very few contact points and as a result even though the horizontal projection indicates a connection between these two lines it is a very weak connection. The solution is to consider the tail of the horizontal projection which holds all the necessary information for segmenting the lines properly yet solves problems of weak line attachment. If the potential lines indicated by Divide & Conquer Level 1 conform to our idea of line consistency then they are segmented and returned. otherwise we continue on.

Figure 2.3.10 illustrates the first level of Divide & Conquer.

8b) The next attempt is based on the observation that if an address block image is divided into several sub-parts each of these sub-parts can have its own horizontal projection that will indicate potential lines that cross this division. This information can be used to segment lines that are meshed or connected because of some big blob or logo that wasn't removed. In practice the image is divided up into 8 pieces, each piece is considered to determine how it could be used to segment the lines in this address block then a final decision is made on how to segment. Again the iine width consistency is measured and if it falls within acceptable bounds the segmented lines are returned, otherwise we keep going. Figure 2.3.11 is a good example of the application of this approach.

8c) The final level of Divide & Conquer deals with images that contain lines that are very intermeshed and difficult to separate. It was noted that if an erosion operation were applied in the North-South direction that the horizontal projection of the resulting image showed some separation between lines and could be used for segmentation. Again consistency measures were taken and if they were deemed to be consistent then the corresponding lines were returned, otherwise the entire image was returned. This is not to say that the segmentation process definitely failed on this particular image - it was sometimes the case that there was only one line in the image and this line was maintained intact as is proper. Figure 2.3.12 shows this final approach.

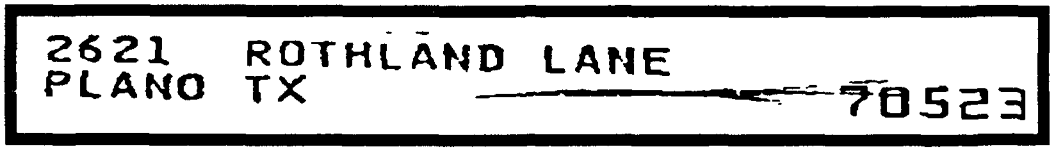
In general, the simple horizontal projection accounts for segmenting ~85-91% of all address block images while divide and conquer 1 accounts for ~3%, with the remainder equally divided between divide and conquer 2 and 3.

9) Finally, there was one address block image anomaly that was easier to detect after the initial segmentation. This problem is referred to as the "Interlaced Text Line" Problem. This is the situation in which the envelope has pre-printed text indicating what information is to follow (for example, "RECIPIENT'S name (first, middle, last)","Street Address" and "City, State and Zip Code"). This can cause severe problems in parsing and later processing because of assumptions about street lines and city-state-zip lines. Figure 2.3.13 indicates the features of these non-informative interlaced text lines that allow them to be detected and removed.

Finally, one rarely occurring, yet important problem is the occurrence of intra-skewed lines, i.e. lines within an address block image that are skewed at different angles relative to each other. We attempt to detect this problem if the address block image fails at the simple horizontal projection level - we can look at the horizontal projection taken for different rotations of the image and make sure that the zero space is maximized. If we find that this is not the case in the original image we can pick out the bad section, deskew it properly and try again. Figure 2.3.14 describes this problem.

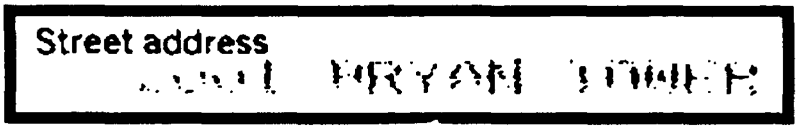
As stated previously the line segmentation module's performance has been increased to a 9o% success rate in Phase II. The algorithm has also proven to be very robust - in a pre-Phase II test on approximately 3500 images the algorithm was successful on over 95% of the images. For the most part the images that we had problems in segmenting were 1) handwritten oı had some writing on them or 2) very poor quality images - either sparse dots or noisy junk.

Why Preprocess Address Block Images Befure Segmentation?



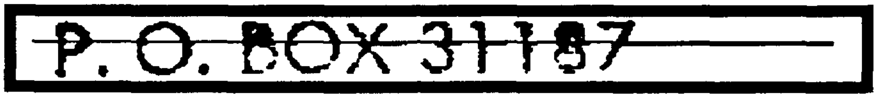
2621 ROTHLAND LANE
PLANO TX 70523

Two lines segmented together because of a slight tilt



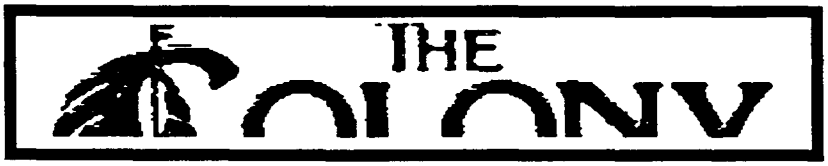
Street address

A non-informative line taken to be part of a good line

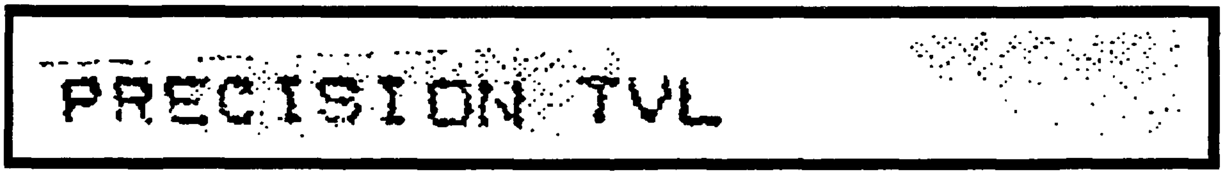


P. O. BOX 31187

A segmented line that will now be nearly impossible to segment into words and characters

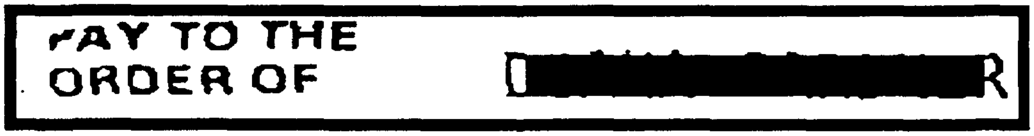


A part of a company logo segmented into a distinct line



PRECISION TVL

A line with noise that distorts the height of the line as well as leading to improper word segmentation

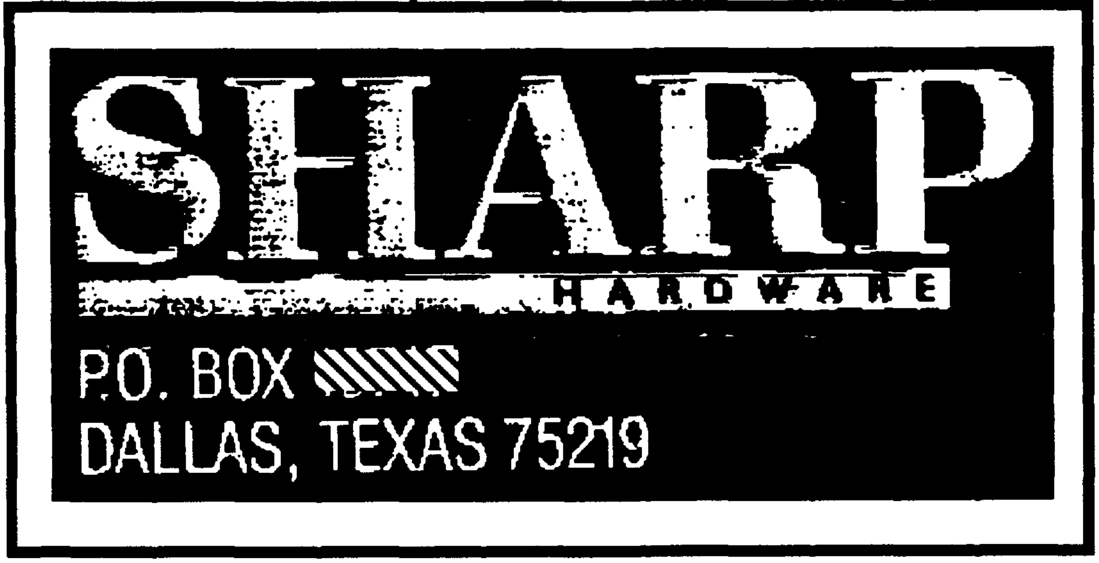


PAY TO THE
ORDER OF [redacted]

A line with the non-informative label included

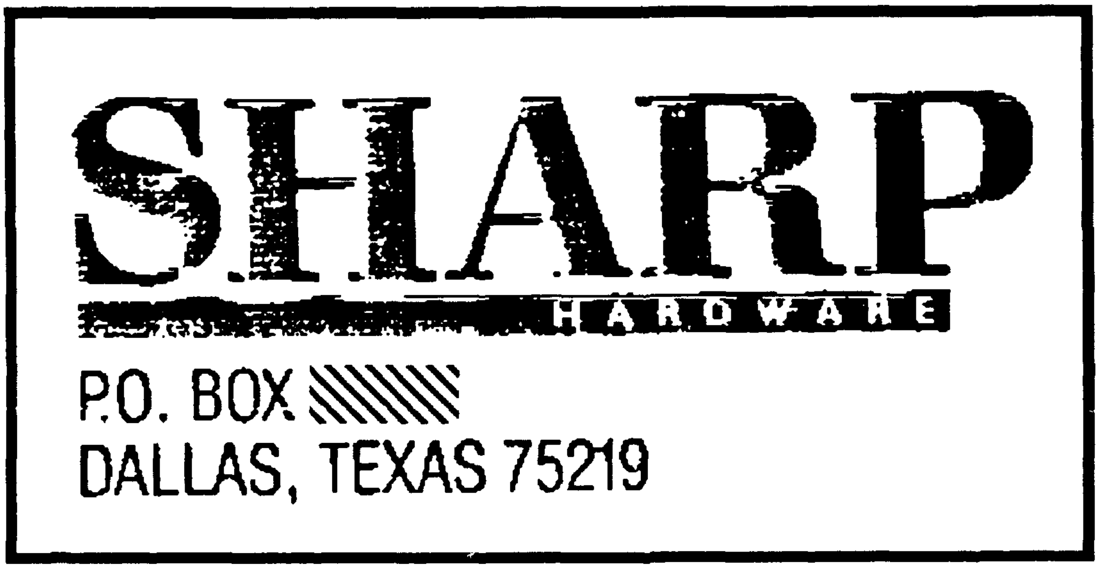
Figure 2.3.2

1. An Example of an Inverted Image



SHARP
HARDWARE
P.O. BOX [redacted]
DALLAS, TEXAS 75219

The original inverted image
(roles of background and foreground reversed)



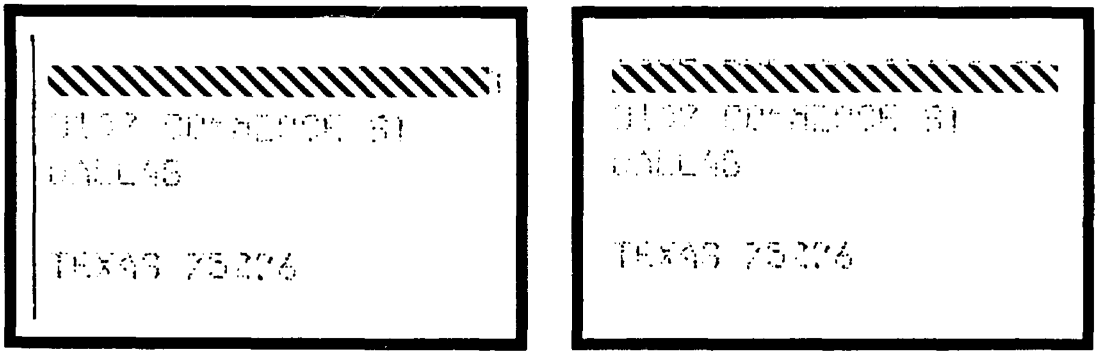
SHARP
HARDWARE
P.O. BOX [redacted]
DALLAS, TEXAS 75219

The complement of the original image that can be segmented

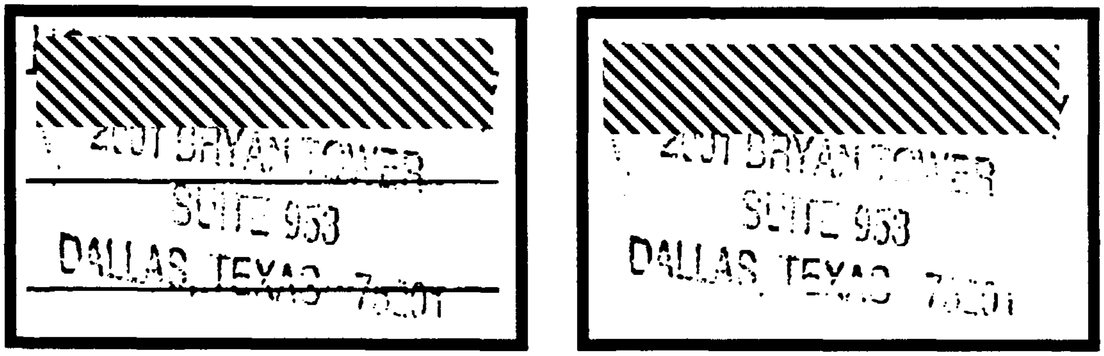
When an image is inverted it is impossible to use our techniques to properly segment the image into lines, words and characters

Figure 2.3.3

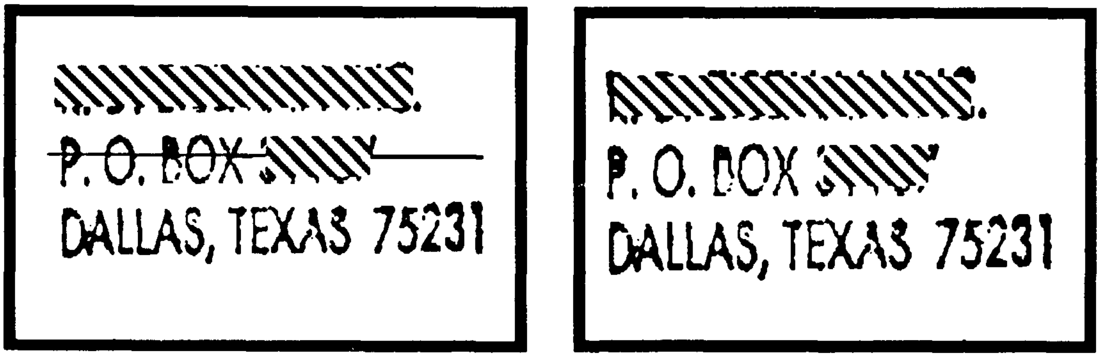
2. Long Horizontal and Vertical Line Removal Using Morphological Operations



Before and after images showing the removal of a long vertical line



Before and after images showing the removal of long horizontal lines

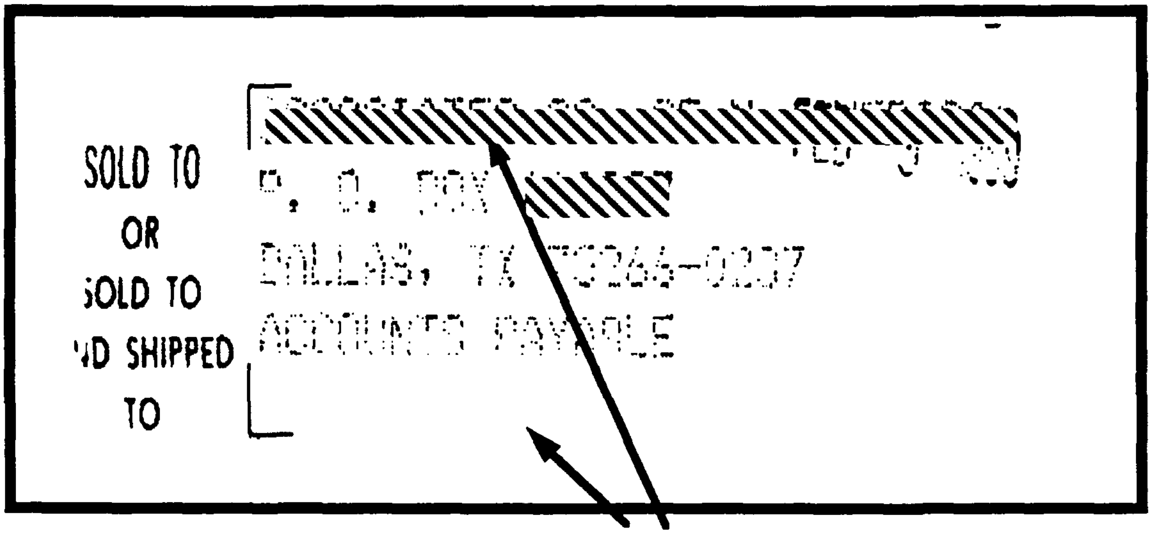


Before and after images showing the removal of a long horizontal line

Long horizontal and vertical lines in the image can cause improper deskewing and can problems in word and character segmentation

Figure 2.3.4

3. Removal of Brackets Using Morphological Operations



SOLD TO
OR
;OLD TO
'ID SHIPPED
TO

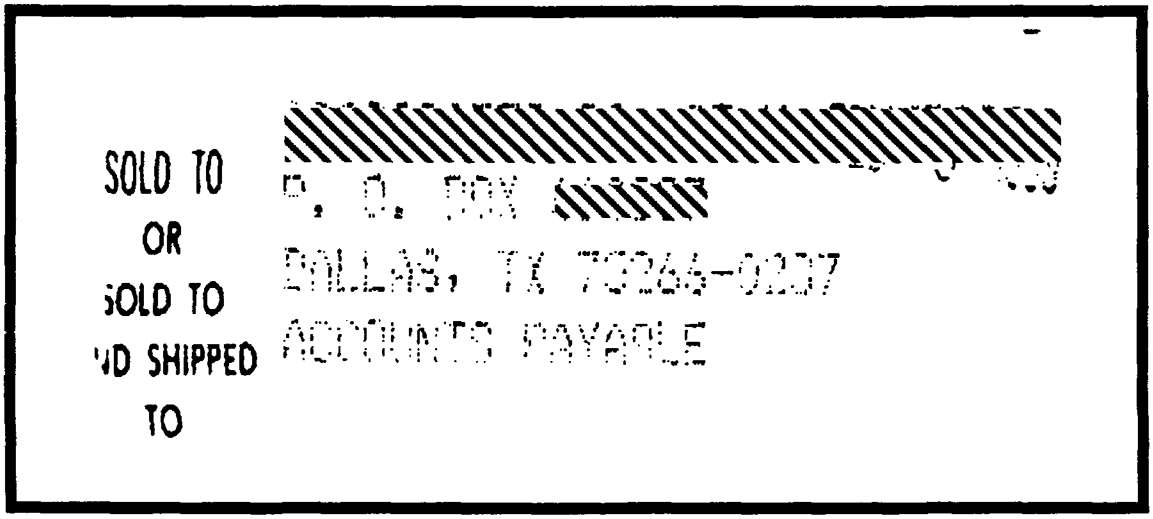
[redacted]

P. O. BOX [redacted]

DALLAS, TX 75264-0237

ACCOUNTS PAYABLE

The image before removal of brackets



SOLD TO
OR
;OLD TO
'ID SHIPPED
TO

[redacted]

P. O. BOX [redacted]

DALLAS, TX 75264-0237

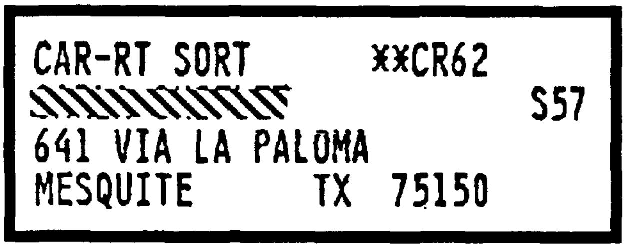
ACCOUNTS PAYABLE

The image after removal of brackets

The large corner brackets present in some image cause problems for line segmentation and word segmentation as well as effecting the characters per word estimate

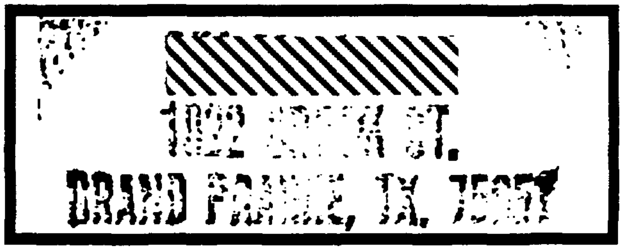
Figure 2.3.5

4. Attempts at Image Quality Enhancement



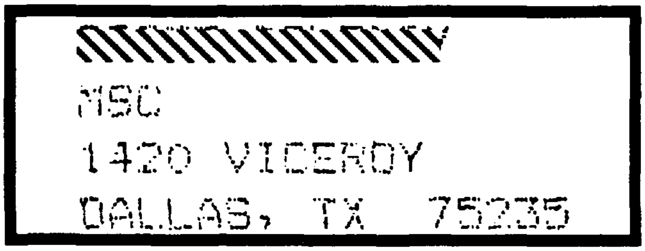
CAR-RT SORT **CR62
S57
641 VIA LA PALOMA
MESQUITE TX 75150

A good quality image.



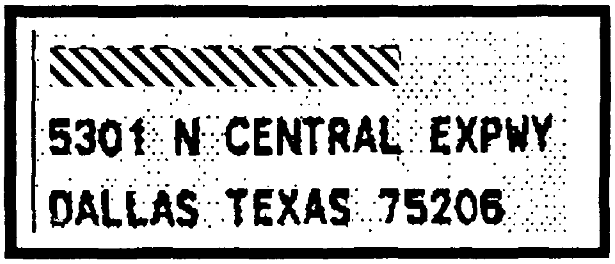
1022 [illegible] ST.
GRAND [illegible], TX [illegible]

A poor quality image due to noise and dropouts



MSC
1420 VICEROY
DALLAS, TX 75235

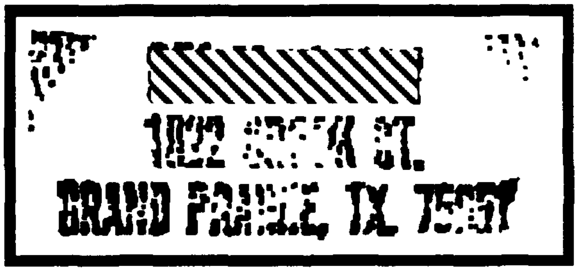
A poor quality image due to dot matrix print



5301 N CENTRAL EXPWY
DALLAS TEXAS 75206

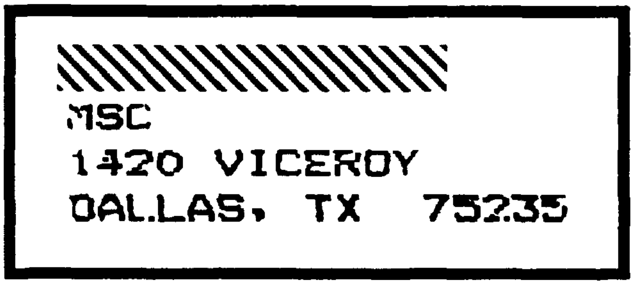
A poor quality image due to spurious noise

Morphological operations have been developed to measure and to attemp to correct some of the quality problems



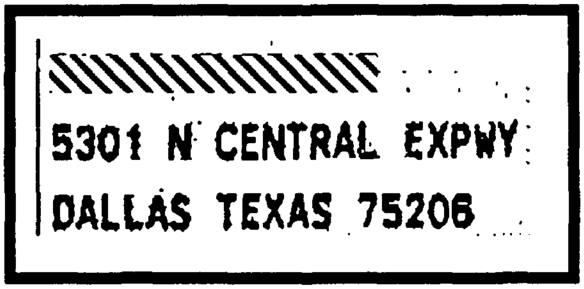
1022 [illegible] ST.
GRAND [illegible] TX [illegible]

The original image fattened for better line segmentation



MSC
1420 VICEROY
DAL.LAS, TX 75235

The original image fattened for better line segmentation

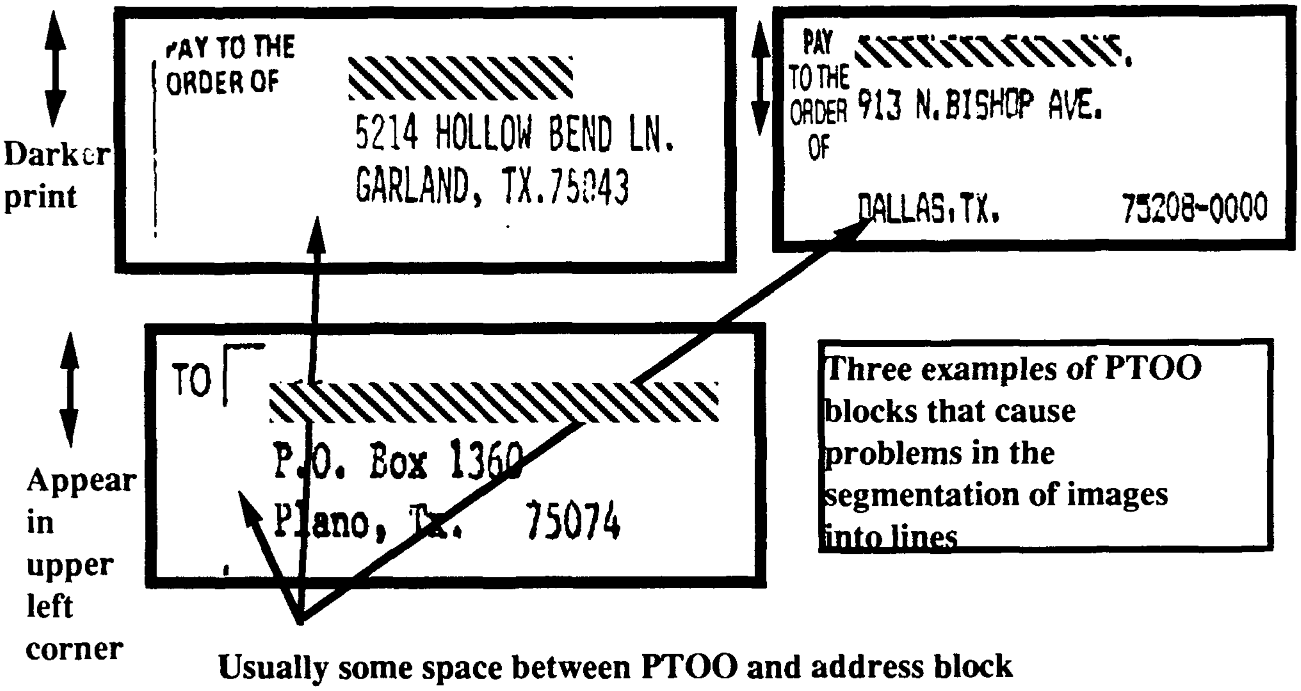


5301 N CENTRAL EXPWY
DALLAS TEXAS 75206

The original image put through a filter to remove dotty noise

Figure 2.3.6

5. Extraneous Blocks of Text in an Image (PTOOs -> Pay To The Order of Blocks)



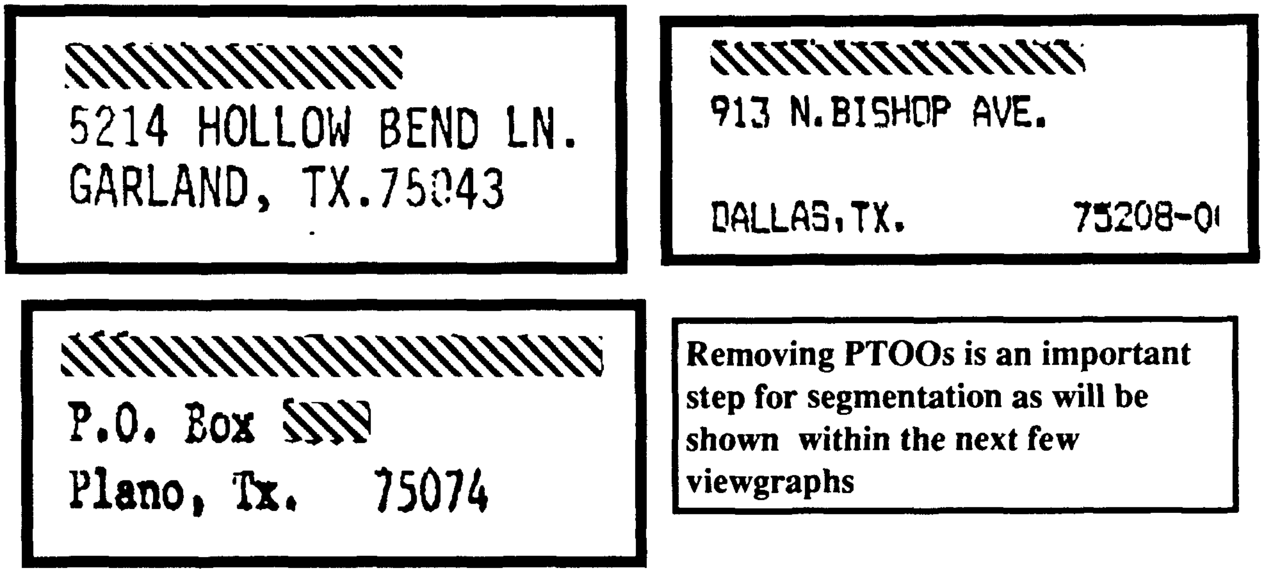
Darker print

Appear in upper left corner

Three examples of PTOO blocks that cause problems in the segmentation of images into lines

Usually some space between PTOO and address block

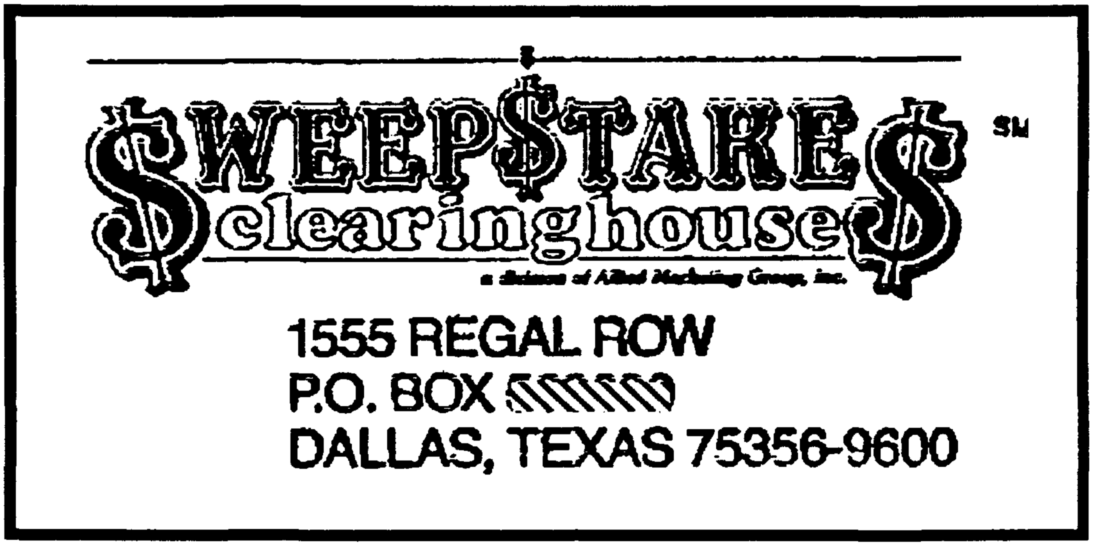
Using characteristics of these type of text blocks it is possible to eliminate them from the address block



Removing PTOOs is an important step for segmentation as will be shown within the next few viewgraphs

Figure 2.3.7

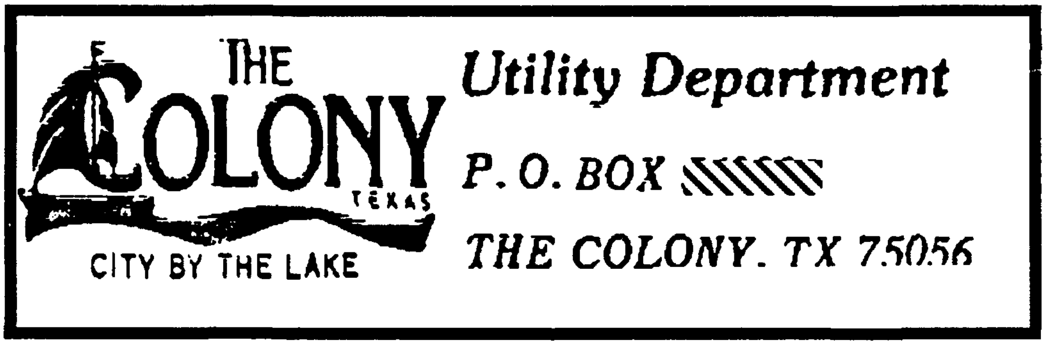
6. Company Logos in Address Block Images



$WEEP$TAKE$ SM
clearinghouse
a Division of Allied Marketing Group, Inc.

1555 REGAL ROW
P.O. BOX [redacted]
DALLAS, TEXAS 75356-9600

A company logo that we attempt to maintain intact as one line



THE COLONY TEXAS
CITY BY THE LAKE

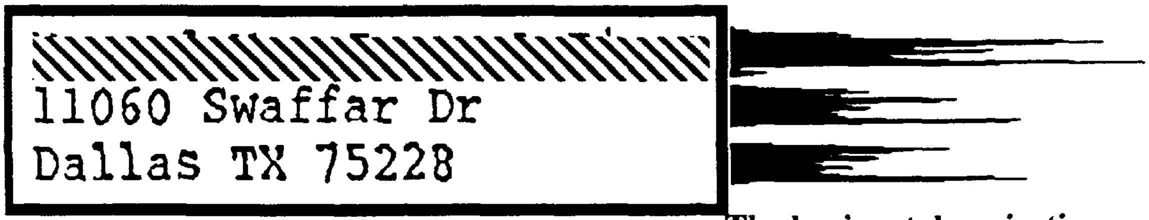
Utility Department
P. O. BOX [redacted]
THE COLONY. TX 75056

A company logo that must be removed using techniques similar to those used for removal of PTOOs

Company logos pose one of two problems:
1) they could be potentially be broken into many lines that would adversely effect parsing
2) they could cause the same problems as PTOOs

Figure 2.3.8

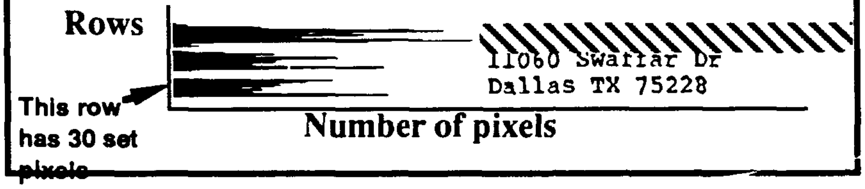
7. Simple Horizontal Projection for Line Segmentation



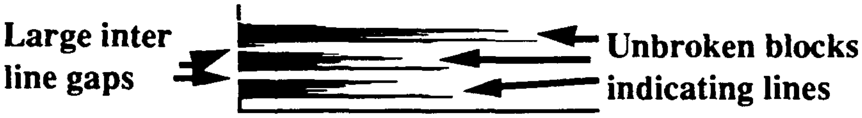
A good quality image

The horizontal projection

The horizontal projection is a histogram of the number of set pixels in each row of the image



In the simple case there are large gaps between the lines in an image and no zero lines within a line



Criterion for simple segmentation:

```
If V(Line Widths) < VarianceThreshold
    Segment Lines
else
    Divide and Conquer
```

From the horizontal projection we can determine the bounding rectangles for the lines and cut them out

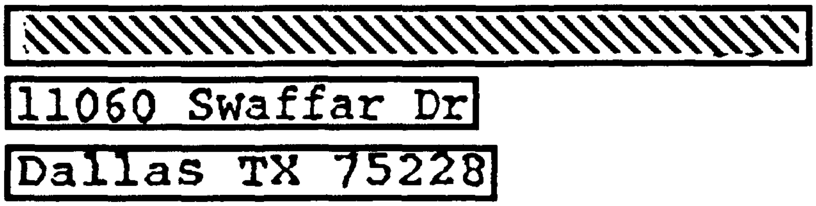
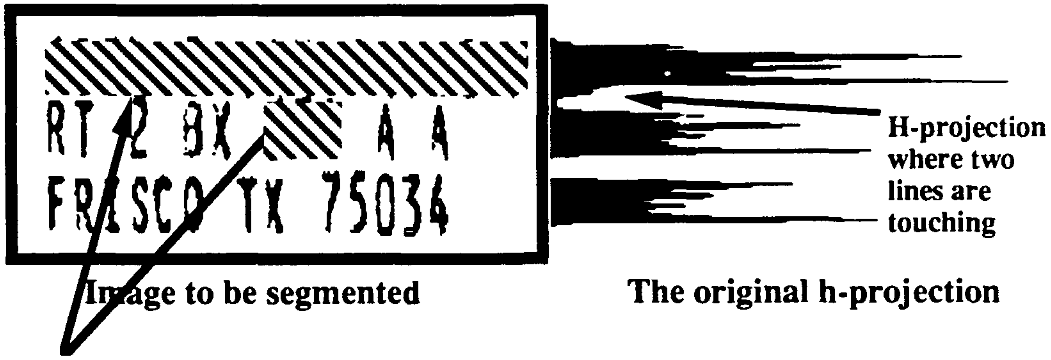


Figure 2.3.9

8a. Divide and Conquer - 1: Use Tail of H-Projection



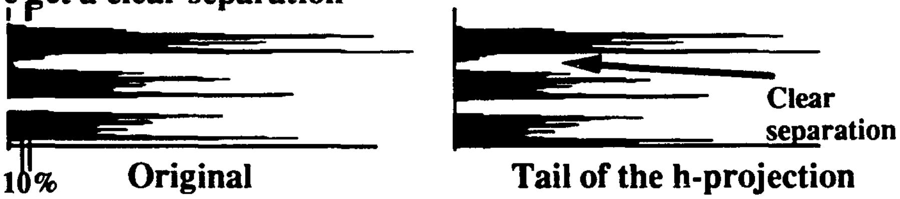
H-projection where two lines are touching

Image to be segmented

The original h-projection

Contact points between two lines that cause them to appear as one line in the h-projection

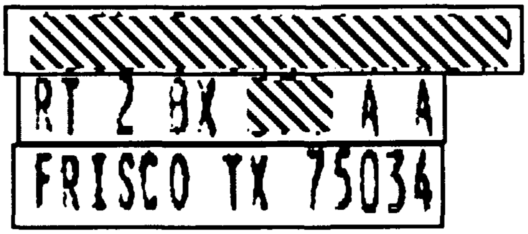
In many cases where two lines appear joined in the h-projection there are very few points at which they touch so that if we take the tail end of the h-projection we get a clear separation



10% Original

Tail of the h-projection

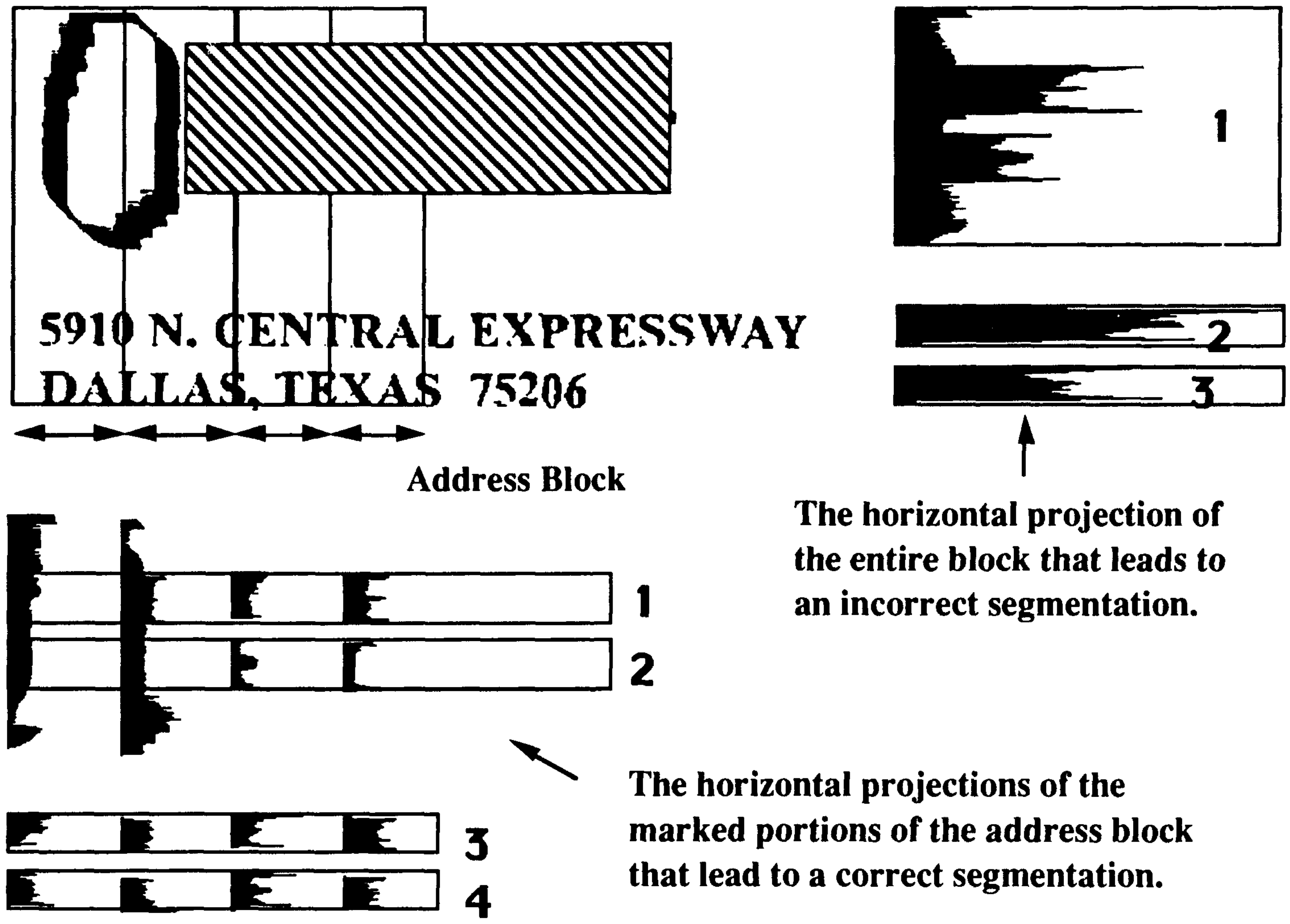
Clear separation



The segmented lines

Figure 2.3.10

8b. Divide and Conquer 2 - Use Partial H-Projections

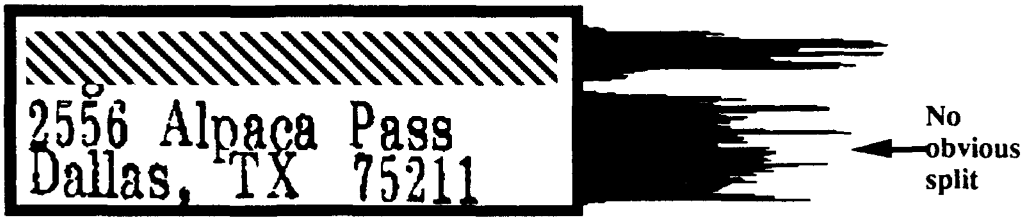


The horizontal projection of the entire block that leads to an incorrect segmentation.

The horizontal projections of the marked portions of the address block that lead to a correct segmentation.

Figure 2.3.11

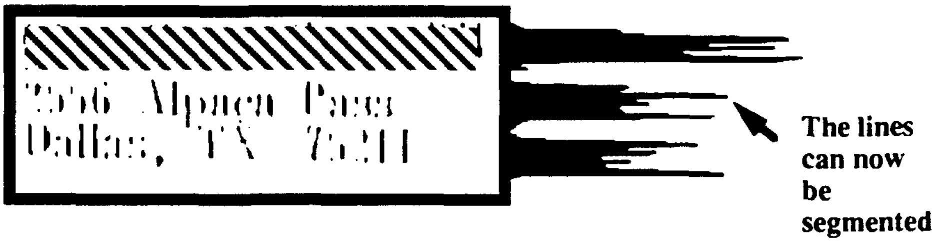
8c. Divide and Conquer - 3 : Use Some Morphology on a Stubborn Image

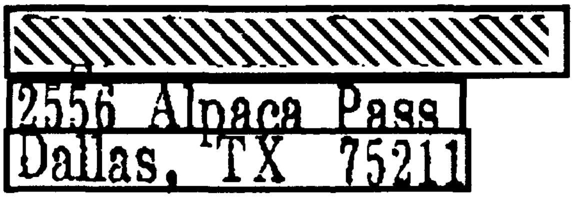


A stubborn image

The original h-projection

In this image there is no easy way to separate the two problem lines. To accomplish this the offending line is eroded North-South using a bar as the structuring element

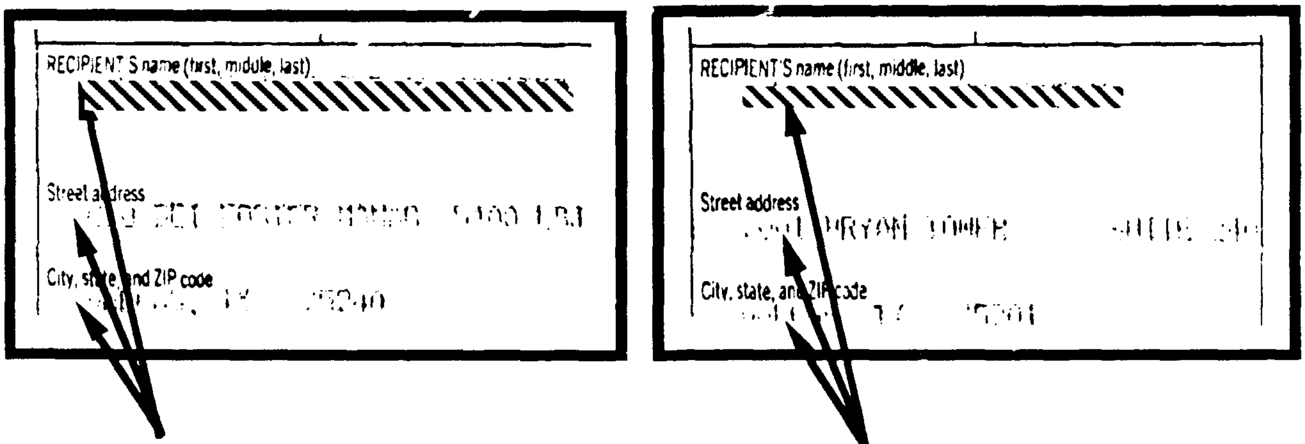




The segmented lines

Figure 2.3.12

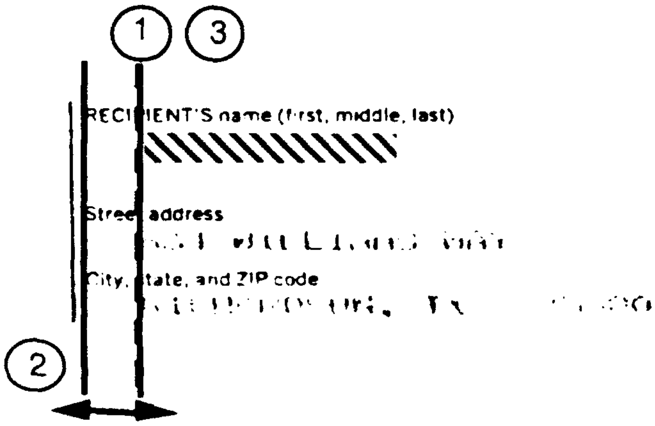
9. Non-informative Lines Interlaced with Text Lines

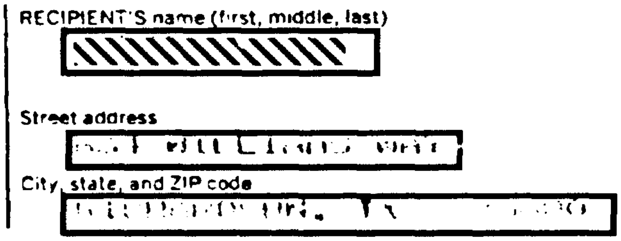

RECIPIENT'S name (first, middle, last)
Street address
City, state, and ZIP code
RECIPIENT'S name (first, middle, last)
Street address
City, state, and ZIP code

Two images with lines that lead to parsing problems, i.e. lines such as "Name", "Street Number","Address",etc.

These lines may be detected after initial segmentation using the following three clues:

1) The non-informative lines are usually in bolder and smaller fonts
2) The Non-informative lines are lined up with each other and the text lines are lined up with each other, but they are not lined up with each other.
3) The Non-informative lines are usually of a better quality.


1
3
2
RECIPIENT'S name (first, middle, last)
Street address
City, state, and ZIP code

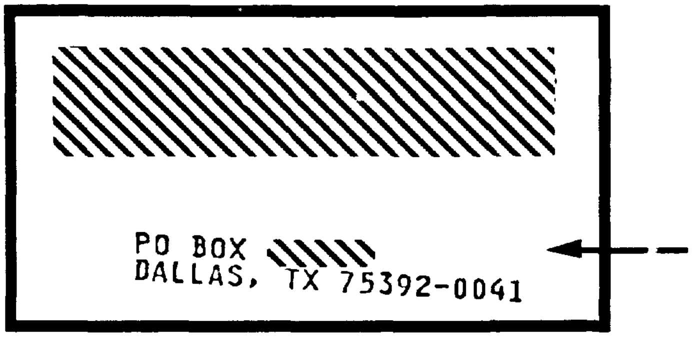

RECIPIENT'S name (first, middle, last)
Street address
City, state, and ZIP code

Lines segmented

Figure 2.3.13

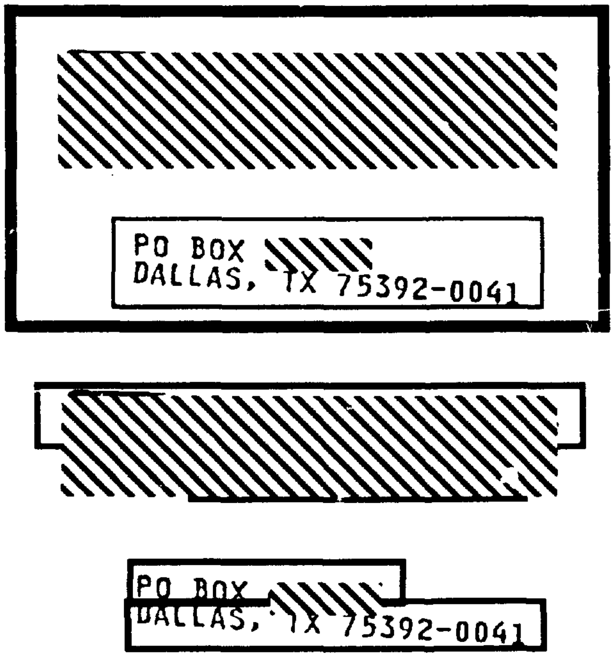
Miscellaneous Line Segmentation Problems

Intra-image skewing occurs very infrequently,but when it does occur it causes major problems in line segmentation



These lines are skewed at a different angle than the first

When this situation is detected and the problem lines are sent through the Hough Transform to be deskewed again



The segmented lines

Figure 2.3.14

2.4 Word Segmentation

Word segmentation operates on a binary image of an address block line and produces a set of word images contained within that line. In Phase II preparation word segmentation received a good deal of attention and two main methods were designed to segment lines into words. Figures 2.4.1 and 2.4.2 present flowcharts for the two main methods in place during Phase II testing.

The first flowchart shows the overall flow of word segmentation. An initial measure of the quality of the line image is taken. Based on this measure a decision is made to use the morphological method or the statistical method of word segmentation. Low quality images are automatically sent to the morphological method because this module has proven to be more effective for this case. As shown in the figure, the statistical method has several checkpoints at which the method may be abandoned in favor of the morphological approach. The morphological method is selected when: 1) the words are too close together, or 2) the estimation of gap size is off, or 3) the variance of gap size is less than the empirically determined value of 30.0, or 4) more than 5 words have been segmented. (There is a low probability of a line containing more than 5 words, and since the morphological method has proven to be more robust we would like this module to process lines of this type).

A very important step in pre-processing of the line images is the attempted removal of punctuation.

As in line segmentation pre-processing, a very conservative approach is taken. The detectors for punctuation (primarily periods and commas) were hand-crafted to use the key characteristics of these marks, i.e. their usual vertical location in the bottom half of a line, their shape, size and lack of cavities, and the appearance of white space before and after. The current detector successfully detects and removes approximately 60% of the punctuation from line images. If cases where there is doubt that a detected blob is punctuation, it is left in the image. Figure 2.4.3 illustrates the method for punctuation removal.

The statistical method for word segmentation is fairly simple. First an attempt is made to estimate the inter character gap sizes. These are then sorted and the variance and median are calculated from which a cutoff value is determined. All gap sizes above the cutoff point are deemed to be inter word gaps and the line is segmented accordingly. If any problems arise, as discussed above, the line image is sent to the morphological method. Figure 2.4.4 demonstrates the statistical method.

The morphological method use morphological operations in an attempt to segment a line into words. First potential words are marked using closing operations. These operations are performed with the structuring element size based on line height, and the nominal character width. Next tall parts of characters are marked (such as the vertical lines in H's etc.). Potential i's, l's and 1's are marked next because these have proven to cause trouble especially for fixed width fonts. These marked characters are then used to make sure that no words are being split improperly. Finally a check is made that no gaps that exceeded the nominal character width are filled in and the lines are segmented. Figure 2.4.5 gives a step by step view of the morphological method. Figure 2.4.6 illustrates some of the problems that have occurred in word segmentation

The word segmentation module's performance has been improved to a 94% success rate up from a 90% rate in Phase I testing. The word segmentation module is responsible for a significant number of system failures, and so will continue to receive attention in the system development.

Word Segmentation Flowchart (Top Level)

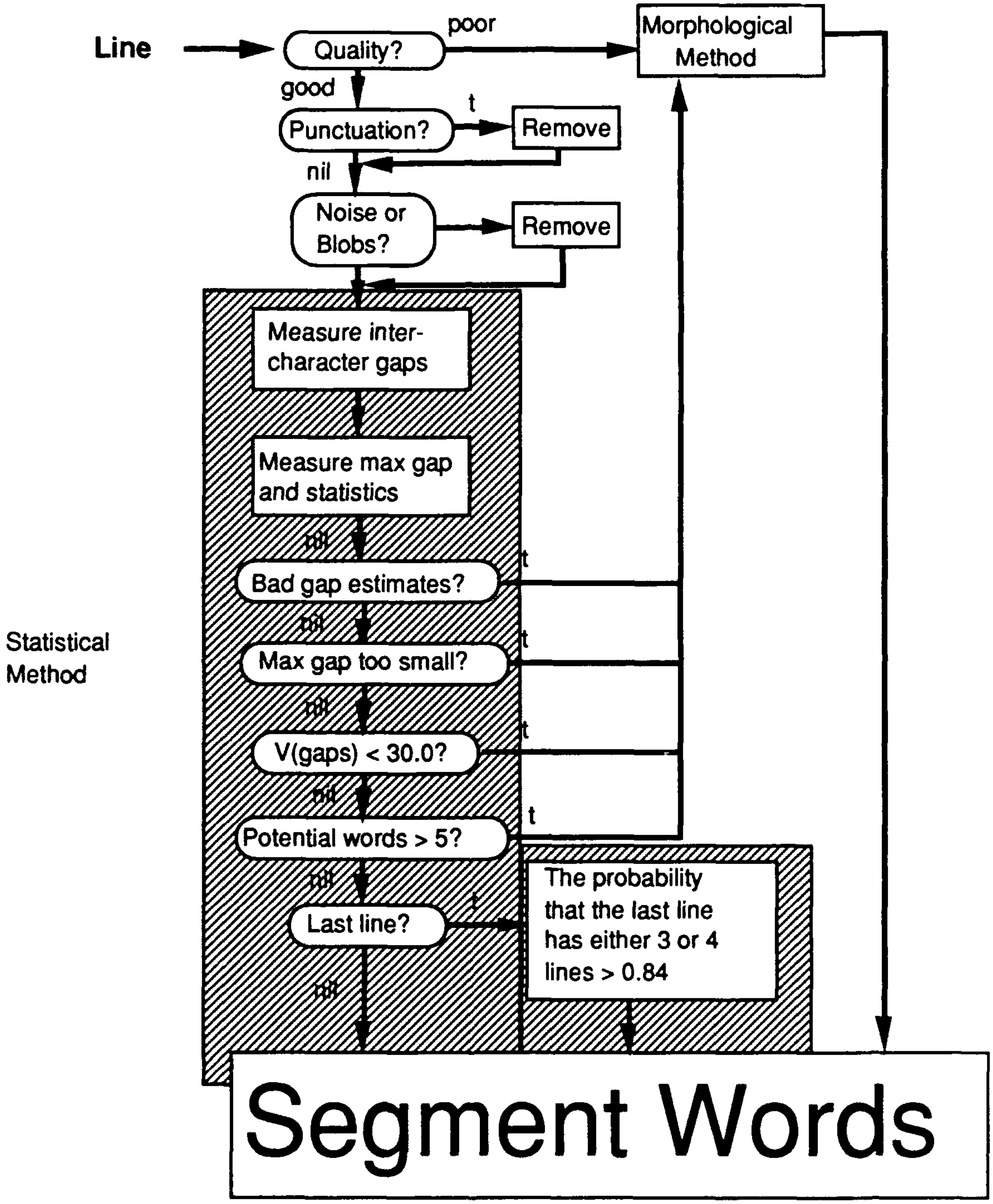


Figure 2.4.1

Word Segmentation Flowchart (Second Level)

Morphological Method

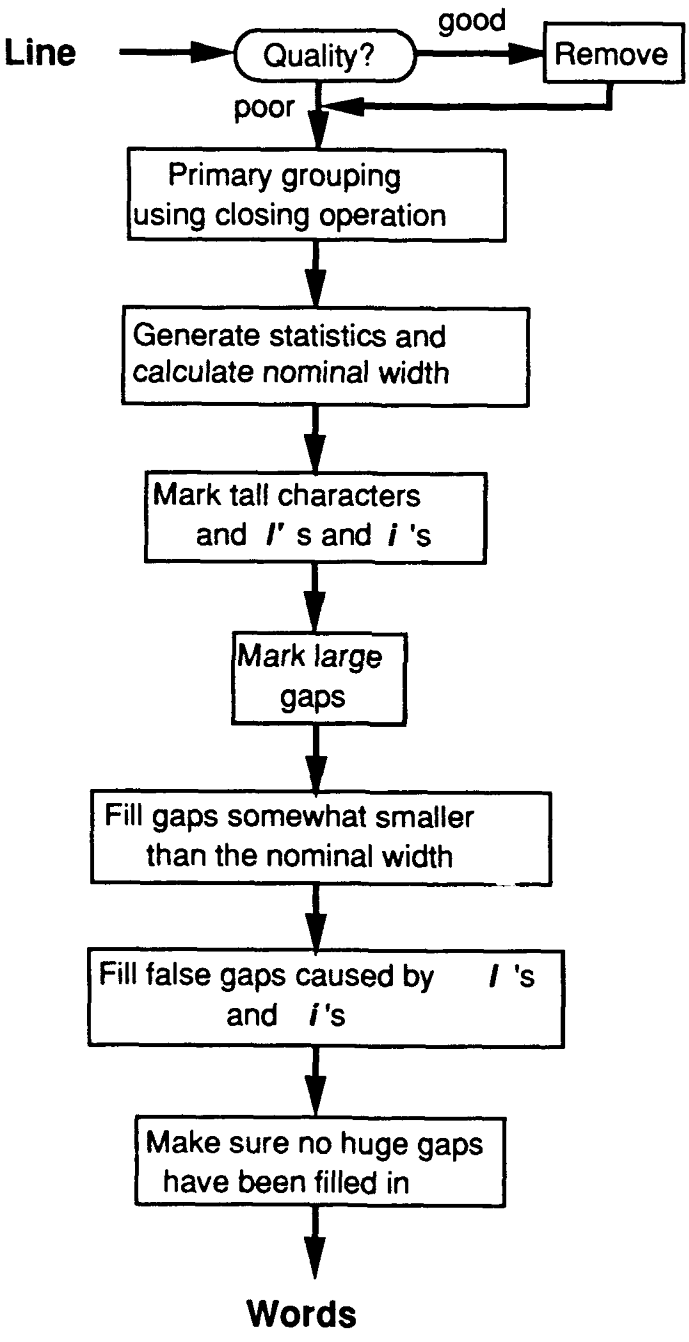
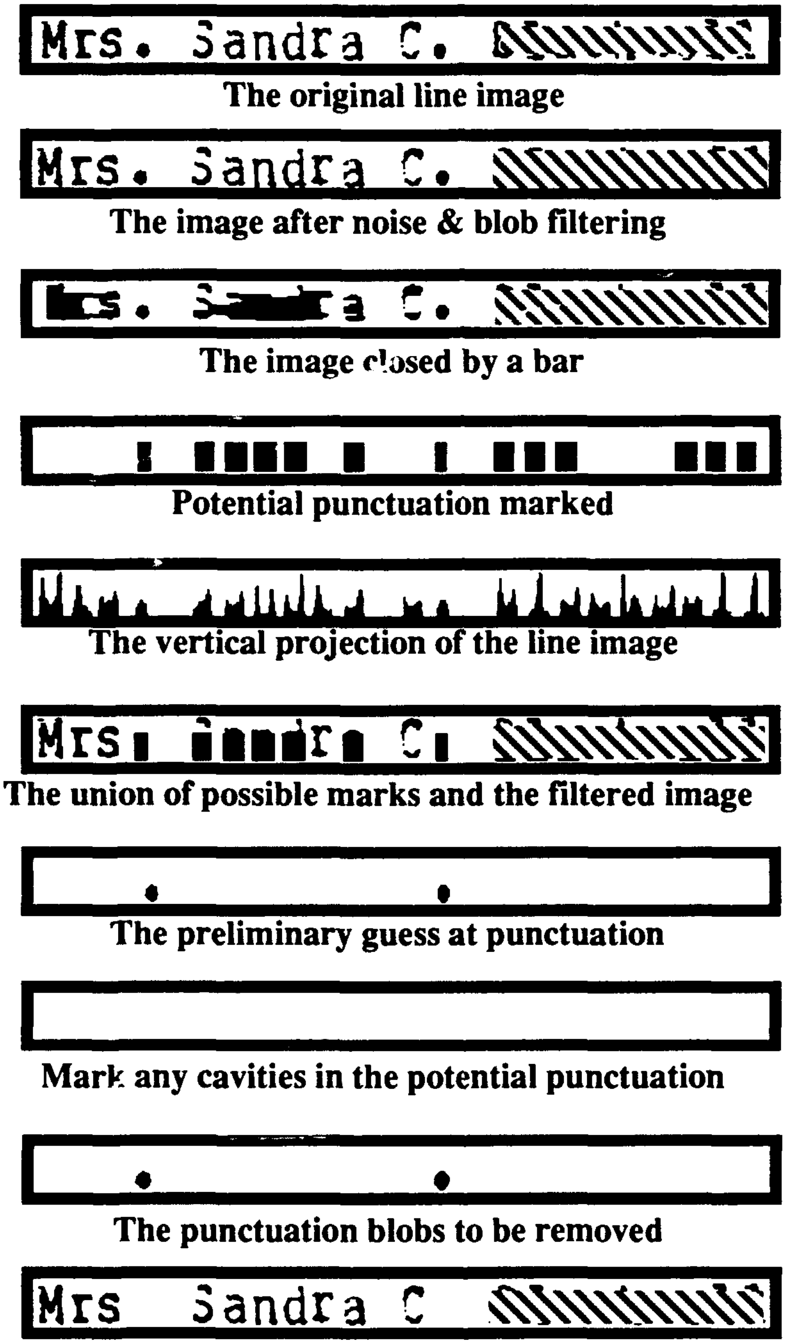

Line
Quality?
good
Remove
poor
Primary grouping using closing operation
Generate statistics and calculate nominal width
Mark tall characters and *l'* s and *i* 's
Mark large gaps
Fill gaps somewhat smaller than the nominal width
Fill false gaps caused by *l* 's and *i* 's
Make sure no huge gaps have been filled in
Words

Figure 2.4.2

Punctuation Removal Before Word Segmentation



The original line image

The image after noise & blob filtering

The image closed by a bar

Potential punctuation marked

The vertical projection of the line image

The union of possible marks and the filtered image

The preliminary guess at punctuation

Mark any cavities in the potential punctuation

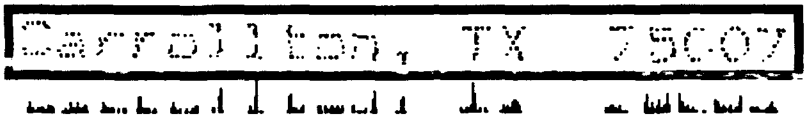
The punctuation blobs to be removed

The original image with punctuation removed

Figure 2.4.3

The Statistical Method for Word Segmentation

Given a line image and its vertical projection:



The inter-character gaps are estimated

The length of each gap is collected into a list which is then sorted in ascending order

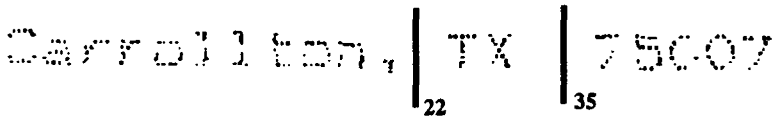
(3 3 3 3 4 4 5 5 5 6 6 6 8 9 10 22 35)

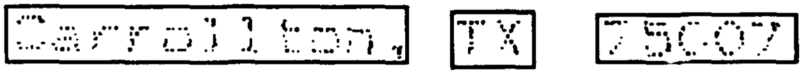
The maximum gap size is determined and a cutoff point is determined using the variance and the median of the gapsizes

(3 3 3 3 4 4 5 5 5 6 6 $\boxed{6\ 8}$ 9 10 22 35)

Median = 5
Variance = ?

In this example the cutoff point was determined to be 15. The line is then segmented by allowing all gaps greater than the cutoff to be inter-word gaps

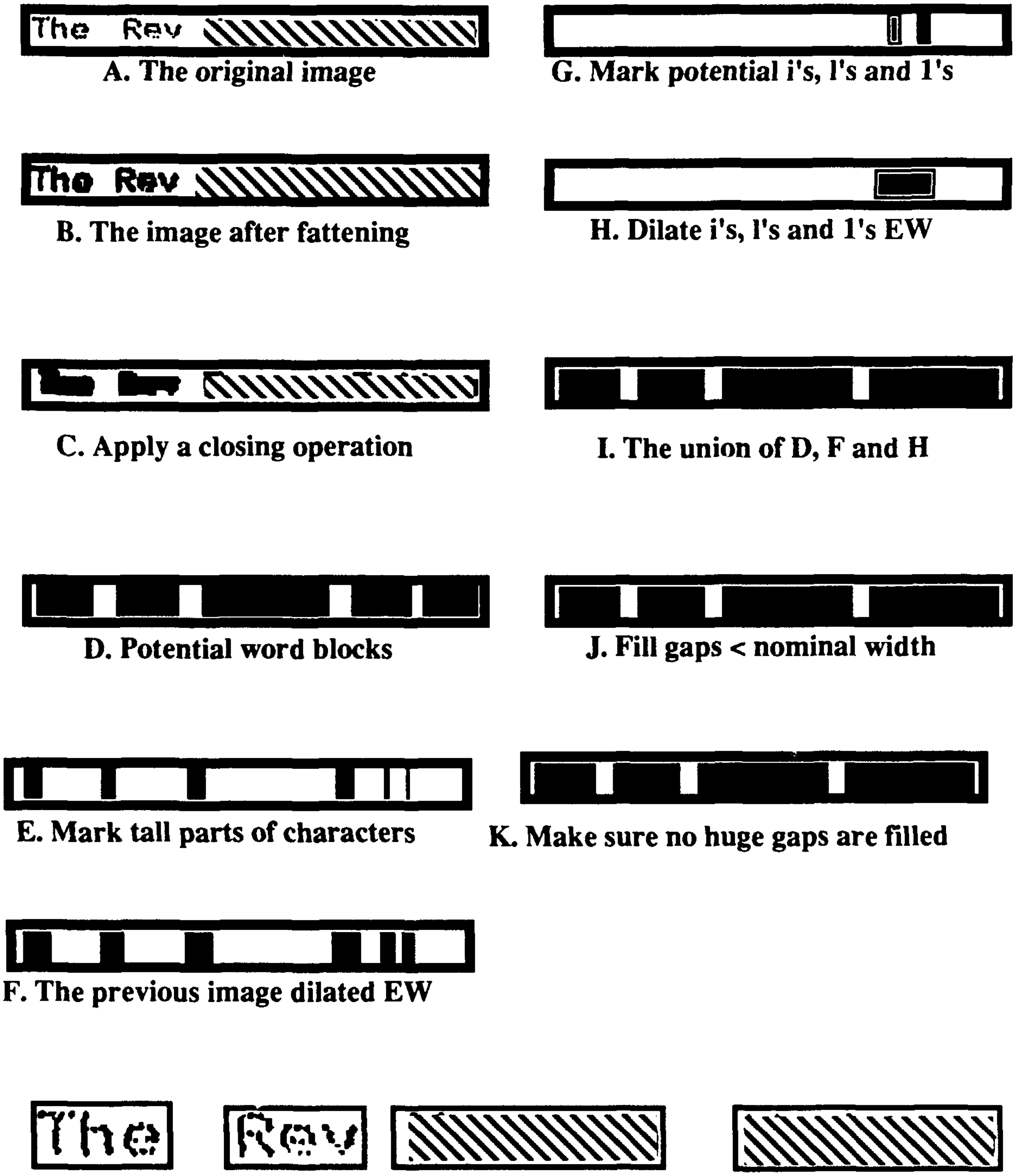




The segmented words

Figure 2.4.4

Morphological Method for Word Segmentation



A. The original image

G. Mark potential i's, l's and 1's

B. The image after fattening

H. Dilate i's, l's and 1's EW

C. Apply a closing operation

I. The union of D, F and H

D. Potential word blocks

J. Fill gaps < nominal width

E. Mark tall parts of characters

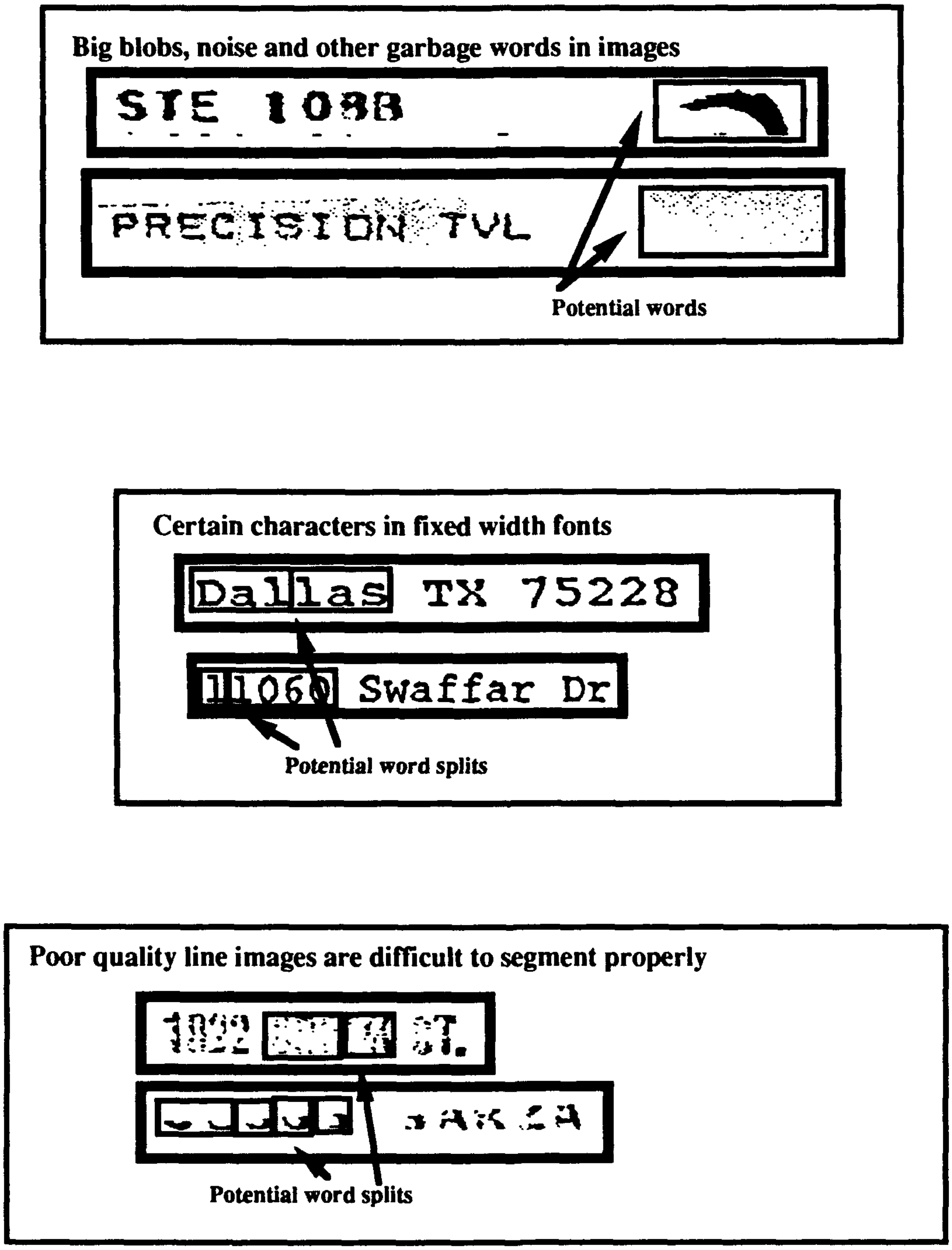
K. Make sure no huge gaps are filled

F. The previous image dilated EW

The segmented words

Figure 2.4.5

Problems for Word Segmentation



Big blobs, noise and other garbage words in images

Potential words

Certain characters in fixed width fonts

Potential word splits

Poor quality line images are difficult to segment properly

Potential word splits

Figure 2.4.6

2.5 Character Segmentation

Figure 2.5.1 is a flowchart for the character segmentation module.

The character segmentation module is responsible for dividing a word image into its constituent character images. The steps in processing a word image are as follows.

1) Vertical Projection - The word image is processed to obtain its vertical projection. The projection records the number of 1 bits (black pixels) in each column of the word image. The overall height of characters in the word image is also recorded.

2) Run and Gap Location - The vertical projection is processed to obtain a series of alternating runs and gaps. The width and area (total number of black pixels) in each run is recorded, and the width of each gap is recorded.

3) Long Run Detection - The runs are examined to determine if any of their widths is greater than 1.2 times the character height. These long runs may be indicative of characters which have been run together, and need to be separated.

4) Artificial Gap Generation - If a long run is encountered, then the run data is reprocessed to include artificial gaps at valleys (local minima) which occur in the middle of runs. Subsequent processing uses the alternating sequence of runs and gaps, where some of the gaps may be artificially created.

5) Preliminary Character Segmentation - The run and gap data is processed starting with the first (leftmost) run. This is assumed to be the start of a character. The system sequentially examines each of the following gaps to determine which one is most likely to be the end of the first character. In general this determination is made based on an expected character width, set initially at 0.74 times the character height. The gaps are examined to find the one which defines a character whose width most closely matches this expected width, with the following exception. If a gap having a width of 3 pixels or greater is encountered before the optimally spaced gap, and the total area (number of black pixels) in the runs accumulated so far exceeds 75 pixels, then this gap is taken to signal the end of the first character. This process is repeated until all of the run data has been assigned to a character zone. The resulting data structure is a new series of runs and gaps, where each run is hypothesized to be a single character.

6) Remove Low Area Characters - The character runs generated in the previous step are examined and any which have an area which is less than 0.07 times the median area are discarded. This step usually just removes small dots following the final character in a word.

7) Secondary Character Segmentation - The process in step 5 is repeated with a new expected character width. The new nominal character width is the median of the width of the characters found in the step 5 processing. The low area characters of this process are also removed.

8) Final Character Segmentation - The two sequences of runs found in step 5 and step 7 are compared, and the sequence with the smallest variance in character widths is taken as the final character segmentation.

Figure 2.5.2 illustrates the character segmentation process.

The character segmentation is responsible for a large number of system failures. The process currently in place is biased toward the processing of fixed pitch text. Methods for other categories should also be developed. Also, the character segmentation module should generate multiple segmentation hypotheses in cases where this is appropriate. With the addition of these capabilities, significant improvements in system performance are expected.

Character Segmentation Flowchart

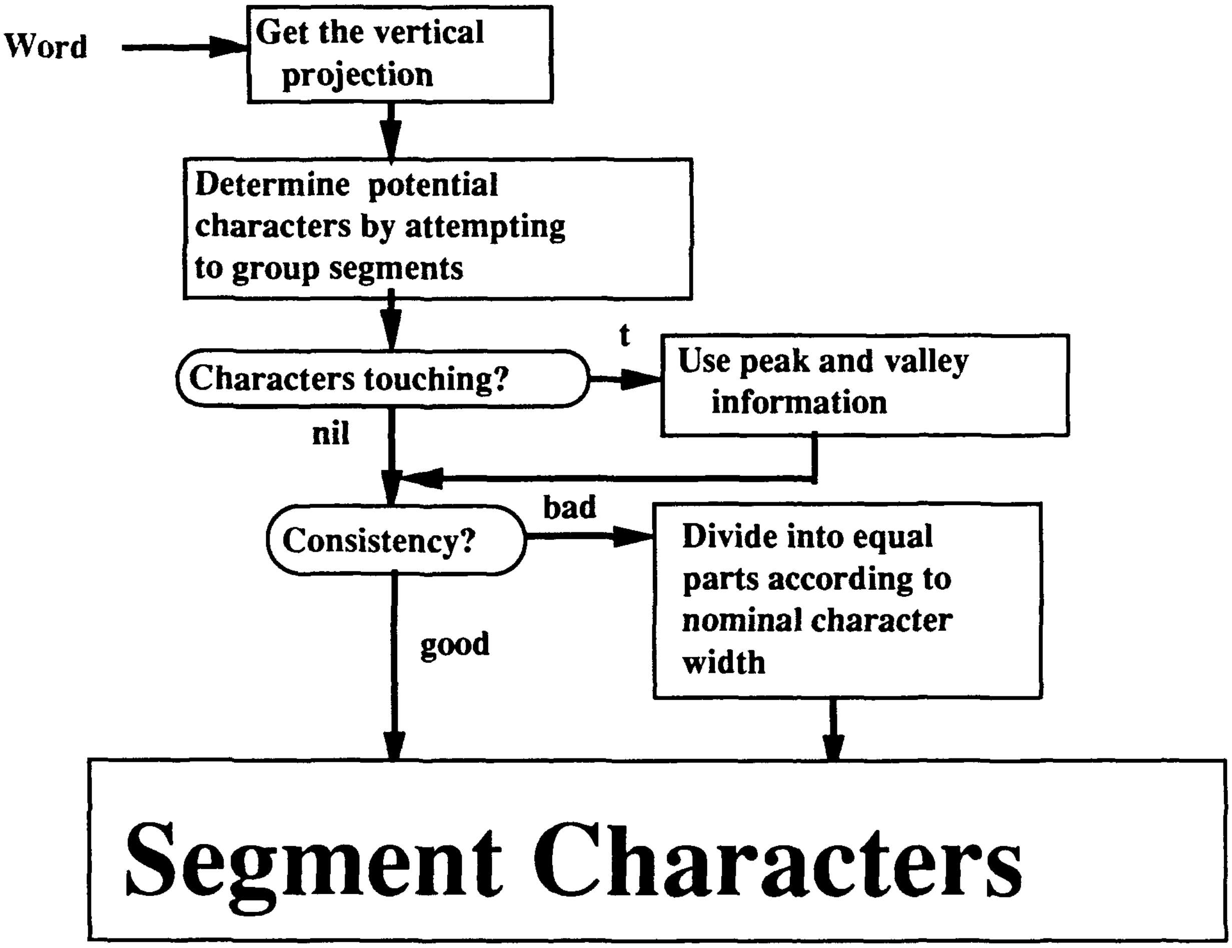
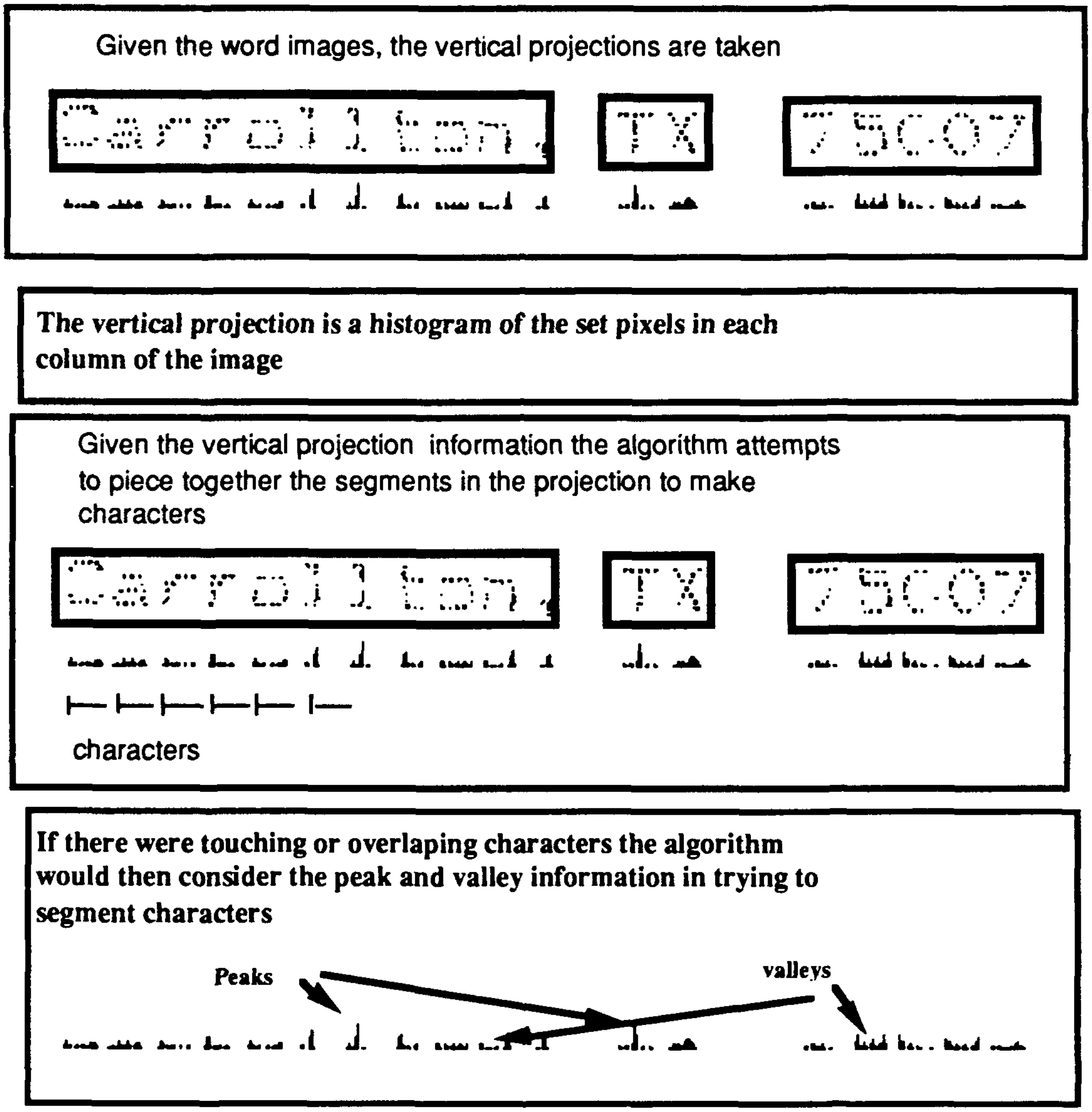

Word
Get the vertical projection
Determine potential characters by attempting to group segments
Characters touching?
t
Use peak and valley information
nil
Consistency?
bad
Divide into equal parts according to nominal character width
good
Segment Characters

Figure 2.5.1

An Example of Character Segmentation



Given the word images, the vertical projections are taken

The vertical projection is a histogram of the set pixels in each column of the image

Given the vertical projection information the algorithm attempts to piece together the segments in the projection to make characters

If there were touching or overlaping characters the algorithm would then consider the peak and valley information in trying to segment characters

Figure 2.5.2

2.6 Word Verification

Word verification is a process whereby a word image is matched against a number of candidate strings. The process assigns a confidence value to each candidate string. Word verification is used in the parsing of address blocks, and also in the matching of street names. This section describes the development of the word verification module. Issues arising during the development process are also discussed.

An important part of word verification is feature detection. Two kinds of feature detectors were used at different stages in the development of the word verification method. One set of features was hand crafted, and another set was generated by an automatic process, using training imagery.

The automatically generated features performed better in the end. Both kinds of feature detectors are described below, as well as their use in word verification.

2.6.1 Word Verification Overview

Figure 2.6.1 illustrates the word verification process.

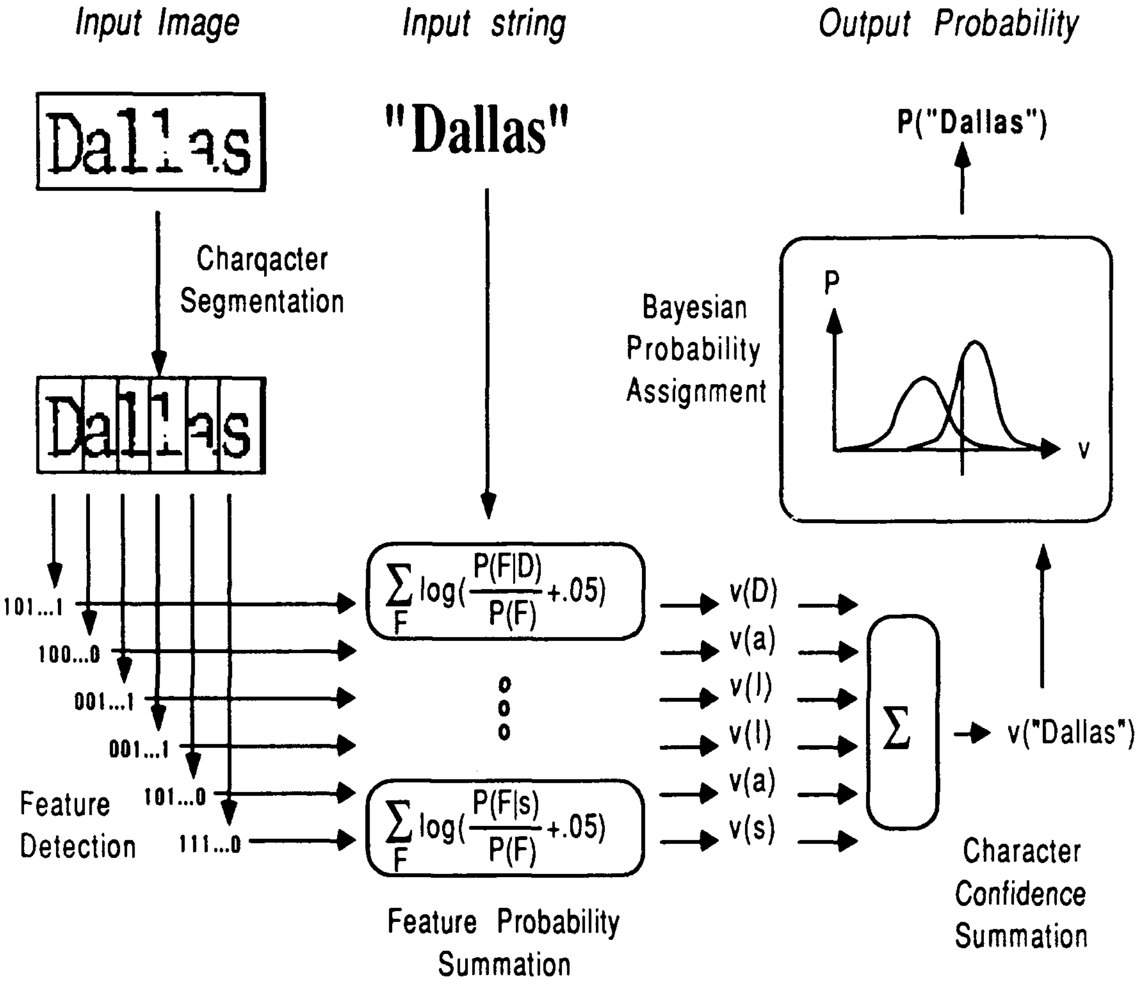


Figure 2.6.1 Word Verification Process

The first step in word verification is character segmentation, as described in the previous section. If the number of characters in the candidate string does not match the number of character zones found by character segmentation, then the process returns 0.0 as the confidence value for that string. If the lengths do match processing continues.

The next step in word verification is feature detection. The feature detectors produce a binary feature detector vector for each isolated character. The feature detectors will be described in a later section.

The next step in word verification is the assignment of a figure of merit to the candidate/image pair. This involves the use of statistics collected from training data of word images paired with their correct ascii strings. For the current system these statistics were gathered from a dataset totalling approximately 67,000 characters. For these character samples, the number of each of the features occurring in each character class was recorded, as well as the number of samples of each character.

From these data we calculate estimates of P(FlC), P(F), P($\overline{F}$ lC), and P($\overline{F}$), for each feature F, and each character C. (Note F indicates presence of a feature while $\overline{F}$ indicates absence of that feature.) We then tabulate for each feature F and each character C, merit values

$$V(F,C) = \log_{10} \left(\frac{P(F|C)}{P(F)} + 0.05\right)$$

and

$$V(\overline{F},C) = \log_{10} \left(\frac{P(\overline{F}|C)}{P(\overline{F})} + 0.05\right) .$$

The merit value for a given word image and character pair is obtained by taking the sum of the merit values for every feature in every zone. The character for the zone is, of course, the corresponding character from the candidate ascii string. If the indicated secondary feature is detected in the zone then the value of V(F,C) is entered into the sum. If the secondary feature is not present in the zone then the value V($\overline{F}$,C) is entered into the sum. Thus, for a word having N characters, the figure of merit is a sum of 47 x N individual character/feature merit values.

The final step in the word verification process is to assign a confidence value based on the derived figure of merit. This is done based on statistics for words matched to their correct strings and to other strings. The statistics are kept separately for each word length (number of characters).

The word verification statistics were gathered from the same word images used to generate the character/feature statistics. The figure of merit for each word image was computed for the correct string for that image, and for all other strings of the same length drawn from a specified lexicon. The mean and variance of the figure of merit was computed for each word length, and for two separate groups: 1) correct strings, and 2) incorrect (other) strings. The probability density of figure of merit V for a given word length and group is estimated by assuming a normal (gaussian) distribution. The final confidence (probability) for a given merit value is computed according to Bayes formula:

$$P(\text{correct}|V) = \frac{P(V|\text{correct})\ P(\text{correct})}{P(V|\text{correct})\ P(\text{correct}) + P(V|\text{incorrect})\ P(\text{incorrect})} \cdot$$

The values P(Vlcorrect), and P(Vlincorrect) are given by the gaussian distribution for the associated word length. The priors P(correct) and P(incorrect) were set heuristically. They are 0.1 and 0.9 respectively.

The confidence (probability) for a given string matched against a given word image can be used to rank the strings for the given word image. The rank of the correct string is a measure of the success of the word verification process. If the correct string is ranked higher than all other strings, then the process can be said to have recognized the word with respect to that lexicon. The confidence value is used in several ways in the contextual analysis system to rank hypotheses, and make decisions about the course of processing.

2.6.2 Manually Generated Feature Set

In the earlier phases of this project, a set of feature detectors for word verification was generated manually. Each of the manually generated feature detectors locates a specific stroke type within a given zone of the character frame. The detectors are based on an alphabet of 24 stroke parts. Each stroke part is described by a morphological structuring element. The size of the structuring element is scaled to the size of the frame containing the character being analyzed. The features are detected by eroding the image by the given structuring element, and then looking for a cue in a specified zone of the character frame. The zones within the frame of each character are localized in the vertical dimension, but not in the horizontal dimension. Thus there are three horizontal stroke part detectors, for the top middle and bottom of the character frame. For the vertical stroke detector, on the other hand, a cue at any horizontal position is taken as a detection. The 24 stroke types give rise to 47 localized feature detectors. Thus, each isolated character gives rise to a 47 bit feature vector. Figure 2.6.1 illustrates this feature set. The left half of the figure shows the 24 structuring elements. The right half of the figure shows the 47 feature detectors, located within the character frame.

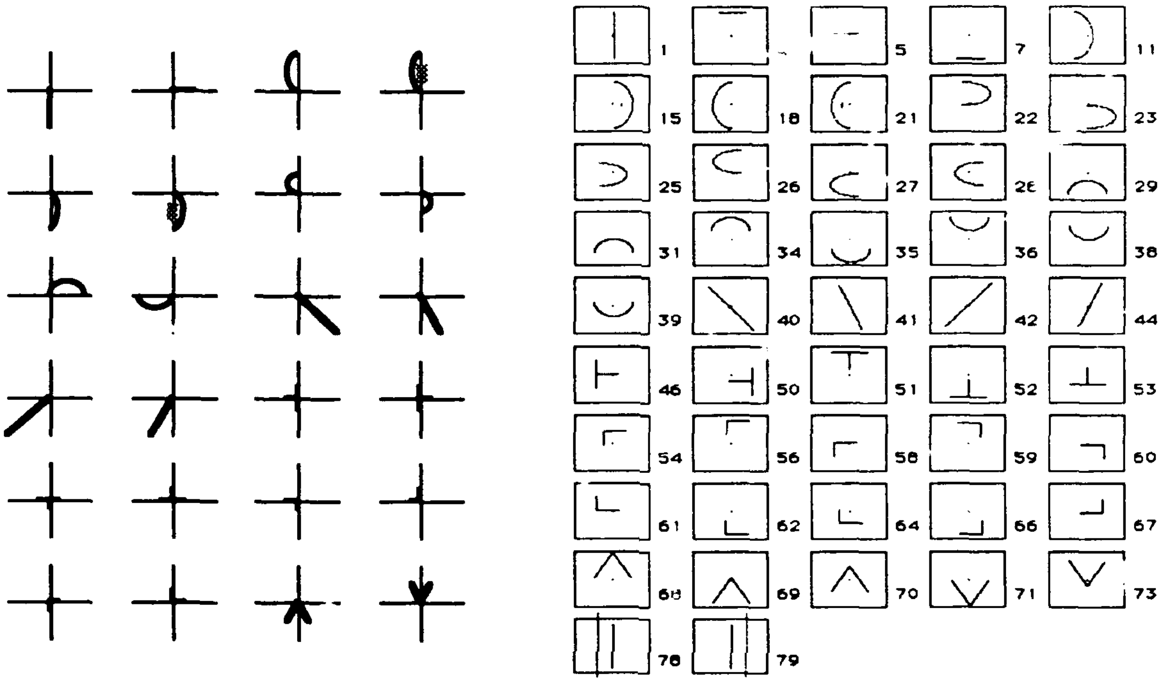


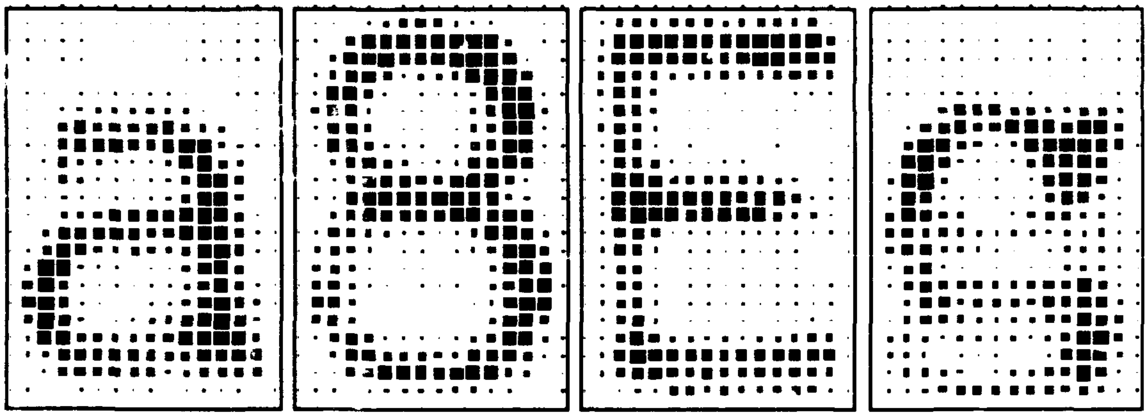
Figure 2.6.2. - Manually Generated Features

2.6.3. Automatic Feature Generation Method

The manually generated set of 47 features represents only one possible feature set of the word verification process. This section describes a process for automatically generating template features for use in the word verification module. The feature generation method is based on the earlier work of Stentiford [6].

The feature generation process begins with a pool of image samples. For this task, we used a training database of roughly 67,000 isolated character images. For the automatic feature generation process and subsequent word verification processes these characters were normalized to a 16x24 grid. The normalization process was a very simple image scaling, with no attempt to keep aspect ratios constant. A random selection of 20 character images for each letter in the character set was made. The character set includes the upper and lower case letters and the 10 digits. The letters "Q", "q", "Z", "z", and "j" were omitted because their representation in the character pool as a whole was too small. This leaves us with 26+26+10-5=57 letters in the character set, and 20 image samples of each letter, for a total of 1140 character images in the training set.

The next step in the process is to form pixel sum images for each letter in the character set. The pixel sum image is a 16x24 array. Each value in the array refers to a pixel location in the character normalizing frame, and counts the number of samples having a black pixel at that location. Figure 2.6.3 shows some example pixel sum images. The images are shown with squares of differing sizes representing the counts at each location in the character frame.



Figu.e 2.6.3 - Example Pixel Sum Images

The pix.l sum images, one for each of the 57 characters in the character set, are used to generate feature detec.ors which are morphological template detectors. Each feature detec.or has a foreground template and a background template, and a zone of detection. Figure 2.6.4 shows a example template feature detector. The feature detectors are generated by a random process as follows.

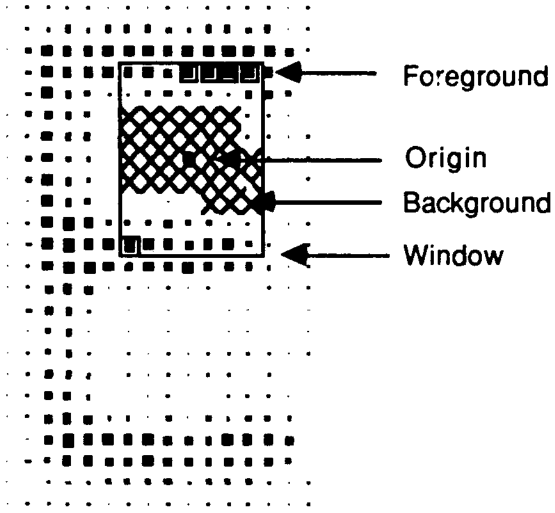


Figure 2.6.4 - Example Feature Detector

One of the 57 pixel sum images is selected at random. A rectangular window is placed randomly within the character frame. The width of the window is uniformly distributed between 4 and 8 pixels (inclusive). The height of the window is uniformly distributed between 6 and 12 pixels (inclusive). The location of the window within the 16x24 pixel frame is uniformly distributed, subject to the constraint that the entire window is within the frame.

The randomly sized and placed window indicates an area within which a thresholding operation will be performed. Those values in the selected pixel sum image which fall at or above the HI threshold become pixels in the foreground template. Those values falling at or below the LO threshold become pixels in the background template. The HI and LO threshold limits are set at 90% and 5% of the maximum value in the pixel sum image respectively.

The foreground and background templates generated by the above process have their origins at the center of the randomly placed window. A zone, 3 pixels wide and 5 pixels high, centered at that origin, represents a detection zone where the feature detector looks for matches. The feature detector is applied to a character image, and either matches or doesn't. Only matches within the detection zone are counted. In morphological terms, a match occurs if:

$$(Z \cap (I \,\Theta\, F) \cap (\overline{I} \,\Theta\, B)) \neq \varnothing$$

where: Z = dctection zone
I = input character image
F = foreground template
B = background template

For purposes of evaluation we identify each feature detector with its vector of responses. This vector has one component for each character in the character set (i.e. 57 components). For a given character, c, the component of the vector is given by:

$$V(f,c) = \frac{N_c(f) * 2}{N_c} - 1$$

where: V(f,c) = component for character c for feature f
$N_c(f)$ = number of samples of c matching f
N_c = number of samples of c, total

This value ranges between 1 and -1. It achieves a value of 1 when all of the feature samples for a given character match the given feature detector. It achieves a value of -1 when none of the image samples for character c match the feature detector.

For our experiments the 20 images which were used to generate random templates were also used as samples for measuring the response of feature detectors. In this case the value for N_c is always 20.

Using these response vectors, the feature detectors are evaluated by a process which measures their orthogonality. A given candidate feature f, is measured with respect to some set of existing features F, by the following formula:

$$M(f,F) = \sum_{f' \in F} \frac{(f \cdot f)(f' \cdot f')}{(f \cdot f')^2 + \varepsilon}$$

where: M(f,F) = the measure of f with respect to the set F
f = response vector for the feature detector
F = a set of feature detectors
f' = response vector for features in the set
$\varepsilon = 0.01$, (a parameter)

The measure is larger for "better" feature detectors. The dot product (squared) in the denominator measures the angle between the two vectors. If they are orthogonal (angle = 90°), this dot product will be zero, thus maximizing the expression. The parameter $\varepsilon = 0.01$ is present to keep from dividing by zero in the case of orthogonal response vectors. The dot products in the numerator normalize the angle with respect to the length of the response vectors. In general this expression increases when response vector of the f feature is orthogonal to others in the set.

A process for generating feature detectors which are "good" according to this measure is as follows. First a set of 60 randomly generated feature detectors are generated. Each of these detectors is measured with respect to the 59 others. The feature detector with the lowest (worst) measure is a candidate for replacement. A new feature detector is randomly generated. It is measured with respect to the 59 not worst feature detectors in the set. If its measure is higher than the worst feature detector, then the worst one is discarded, and the new one is added to the set. All 60 are now evaluated again with respect to the 59 others. Their new values are only different in one term of the sum, that corresponding to the newly added detector. Still, this may change their relative rankings. The worst of this set of 60 is now a candidate for replacement. This process is repeated as long as desired.

The process described above was run starting with 60 feature detectors. The cycle of test and (conditionally) replace was run 500 times. The 60 feature detectors generated by this process were tested in the word verification module.

The results of this test compared with the results from the original feature set are shown in Table 2.6.1. The test used 356 word images which were correctly segmented into characters. Each image was verified against 5000 randomly selected strings from a lexicon of approximately 9000 strings. The table shows the number of images (and percentages) achieving ranks 0 through 10. The left column of results are for the 47 manually generated features. The right column of results are for the 60 new features generated by the auto-template method. The statistics used in the figure of merit assignment for the two feature sets were gathered from the same set of 67,000 (approx.) characters.

Table 2.6.1 - Comparative Word Verification Results

rank	Original Features		Auto Template Features	
0	298	83.7%	336	94.4%
1	309	86.8%	343	96.3%
2	320	89.9%	345	96.9%
3	324	91.0%	346	97.2%
4	328	92.1%	347	97.5%
5	328	92.1%	348	97.8%
6	329	92.4%	348	97.8%
7	330	92.7%	348	97.8%
8	330	92.7%	349	98.0%
9	330	92.7%	349	98.0%
10	330	92.7%	351	98.6%

The new features led to better performance in the word verification test. The number of images ranked 0 (that is, images in which the correct string was preferred over all other 4999 strings) increased by more than 10% over the original method.

With this encouragement, the auto-template feature generation method was investigated further. The 60 features used in the preceding test, you will recall, were the result of 500 cycles of "evolution" using the orthogonality measure. These 60 features were then subjected to an additional 1000 cycles, for a total of 1500 cycles. During this process a significant number of new features were inserted into the feature set because of their better orthogonality scores. This set of auto-template features representing 1500 cycles of the auto-template process was evaluated in the word verification module. The results of this test, shown in Table 2.6.2, are discouraging in that the less highly "evolved" features (500 cycles) performed better than the more highly "evolved" ones (1500 cycles). This is in spite of the fact that the measure of orthogonality was being improved by the additional cycles.

Table 2.6.2 - Word Verification Performance at 1500 Cycles

rank	performance at 500 cycles		performance at 1500 cycles	
0	336	94.4%	329	92%
1	343	96.3%	338	95%
2	345	96.9%	341	96%
3	346	97.2%	342	96%
4	347	97.5%	344	97%
5	348	97.8%	344	97%
6	348	97.8%	346	97%
7	348	97.8%	347	97%
8	349	98.0%	347	97%
9	349	98.0%	347	97%
10	351	98.6%	347	97%

In an attempt to understand these results, several steps were taken. Firstly an observation was made that the word verification test involves a random element in the selection of 5000 strings for each image. The features from 500 cycles were tested again, and the results changed. In the second test the number of images at rank 0 was 331 (93.0%) rather than 336 (94.4%). Clearly this test leaves something to be desired when comparing feature sets.

Another test based on per character rank was developed. In this test, the characters in the training set (20 characters per class x 57 classes = 1140 characters) are ranked using the response vectors generated for the orthogonality measure. Using this new test, the features at 500 cycles correctly classified 66.4% of the training characters, while the features at 1500 cycles correctly classified 62.8% of the characters. The 1500 cycle features are slightly worse than the 500 cycle features using this measure as well.

2.6.4 A Variation on the Feature Generation Method

A second observation was made concerning the form of the orthogonality measure. In its original form the measure is supposed to increase for better feature detectors. Another form of this measure would be:

$$M'(f,F) = \sum_{f' \in F} \frac{(f \cdot f')^2}{(f \cdot f)(f' \cdot f') + \varepsilon}$$

in which the terms of the sum are essentially inverted. This measure should decrease for better feature detectors. One qualitative difference between the two measurement strategies is that the M measure emphasizes perfect orthogonality to a few other features, while the M' measure emphasizes adequate orthogonality to a large number of features. This is especially true if ε is small, in which case one small value for $f \cdot f'$ can dominate the sum for measure M.

An experiment was run using the new, M', measure. As before, 60 features were generated using 500 cycles of the "evolution" process. These features were used in a word verification test for which 92.1% of the images were at rank 0. This is slightly worse than the 94.4 % obtained for the 500 cycle run using the original M measure, and even slightly less than the 93.0% achieved on the second run of the word verification for those features. The differences in these results given the variability in testing is probably not significant. Using the (repeatable) character rank test, the new features (generated with M') correctly classified 76.0% of the training characters as opposed to the original 66.4% achieved by the 500 cycle, M measure features.

To summarize, the new (M') features are better according to one test (recognition rate on training characters), and worse according to another (word verification rank). The word verification test has several disadvantages. The first is its lack of repeatability, due to the random selection of strings from the lexicon. This could be alleviated by choosing a fixed lexicon, and using all strings for every image. Another disadvantage of this test, however, is the use of lexicon strings at all. The use of a lexicon makes the job of word recognition significantly easier than the character ranking task. The number of useful test cases (i.e. ones where there is a possible confusion) among the 356 test images is probably rather small. Thus the word verification test, as formulated, greatly reduces the effective size of the test set. We need a better predictor of bottom line performance.

2.6.5. Interpretation of Results

In an attempt to qualitatively understand the differences between the feature sets generated by the two orthogonality measures, a method for plotting response vectors was developed. The method generates a plot with 57 columns, one for each character class, and 60 rows, one for each feature. The character classes appear in the following order:

"0123456789ABCDEFGHIJKLMNOPRSTUVWXYabcdefghiklmnoprstuvwxy".

Each entry shows a filled or hollow square of a given size. The large filled squares represent features which match many (or all) of the images in the corresponding class(vector component near 1). The large hollow squares show are for features which match few or none of the image samples in the corresponding class(vector component near -1). The small squares (filled or hollow) represent features which match about 50% of the image samples in the corresponding class(vector component near 0). The ideal feature is composed of half large filled squares and half large hollow squares. Also the locations of the filled and hollow squares are "orthogonal" to the other features in the set. The features are displayed with the worst features (according to the orthogonality measure) at the top and the best features at the bottom.

Another method displays the relative orthogonality between pairs of features in a feature set. Each square in this plot of 60 by 60 squares, represents the cosine of the angle between the response vectors, f and f', of two features. That is:

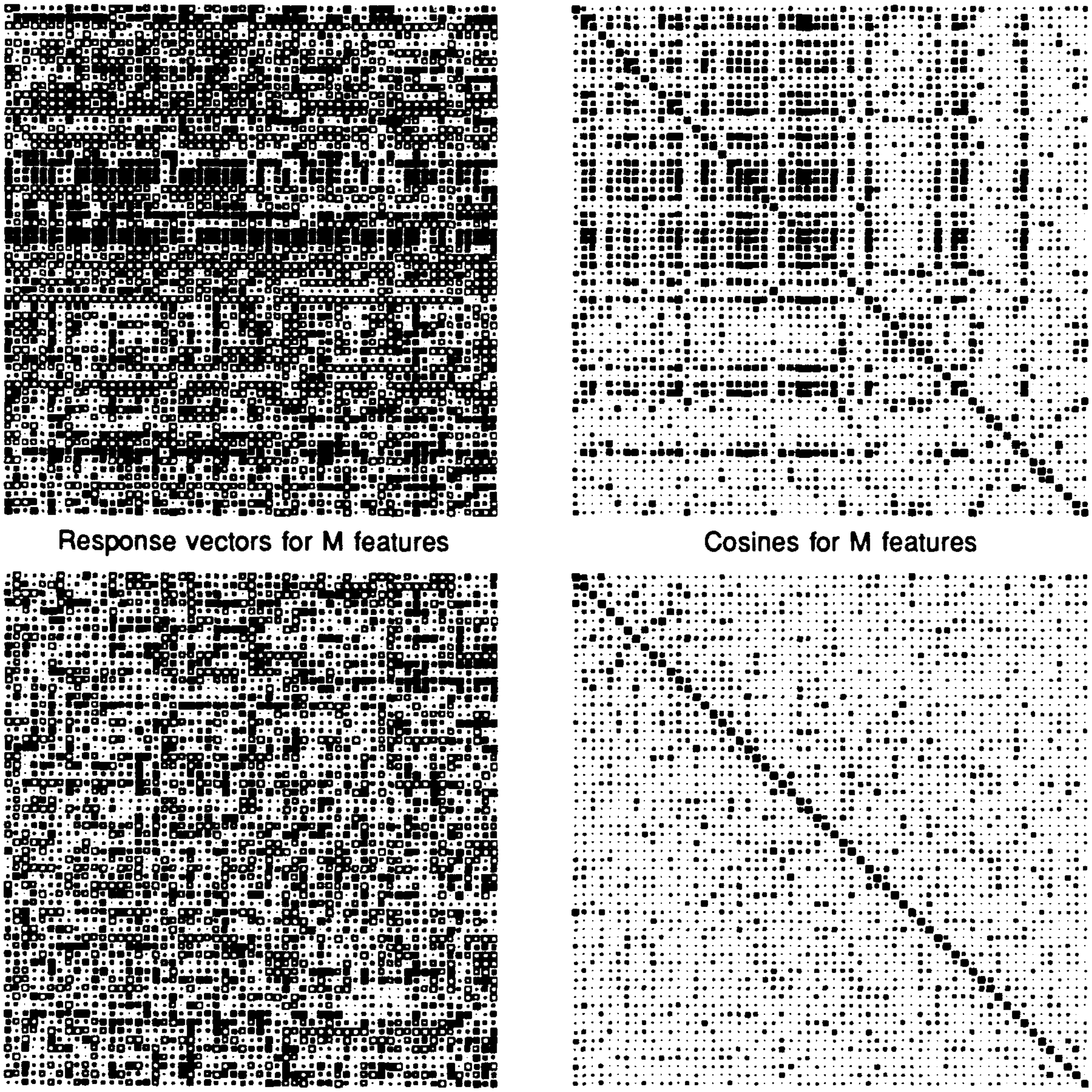
$$\cos^2(\alpha) = (f\cdot f')^2/(f\cdot f)(f'\cdot f')$$

A small square represents a small $\cos^2$ and hence orthogonal feature vectors. A large square, on the other hand, represents features with non-orthogonal response vectors.

Figure 2.6.5 presents plots which compare the features generated by the M and the M' orthogonality measures. These plots clearly show that the M' measure is better at producing orthogonal features.

Another measure of feature quality, however, is feature vector length. Long feature vectors correspond to features which are consistent in their response to each character class. The average length of feature vectors in the M measure features is 6.0 as opposed to 5.0 for the M' feature set (500 cycles). This is compared to a maximum length $\sqrt{57} = 7.55$ for feature vectors for a 57 class problem. Once again, the two measures of feature set performance conflict.

It appears that the orthogonality measures used for these experiments do indeed select useful features for the word verification. The features generated by the automatic method perform better in the word verification test than the manually generated features. The generation process does not, however, continue to improve feature set performance through continued iteration. It is suspected that the orthogonality measure, while continuing to improve, actually results in shorter response vectors, and hence poorer character classification. Experiments using a measure which combines orthogonality with vector length are planned.



Response vectors for M features

Cosines for M features

Response vectors for M' features

Cosines for M' features

Figure 2.6.5 - Vector Plots

2.6.6 Word Verification Test Results

At the end of phase II the word verification module, using the 60 features generated using the M' orthogonality measure, was run on two separate test sets: 1) the city names test set, and 2) the street words test set. The address block images for these two tests were selected from the 929 images included in the Phase II test set. Table 2.6.3 details the images eliminated from the test due to missing or non-alphabetic characters in the city or street word, and word segmentation failures. It also shows the number of word images extracted from the remaining address blocks.

Table 2.6.3 - Word Verification Test Sets

City Test Images		Street Test Images	
invalid city word	14	invalid street name	361
ERIM bad segmentation	65	ERIM bad segmentation	63
SUNY bad segmentation	52	SUNY bad segmentation	56
total address block images	809	total address block images	468
city word images	834	street/suffix word images	850

The results of the city word verification test are shown in table 2.6.4. The table reports results separately for each sortation category (9, 5, and reject) as well as for the total test set. The second column of the table reports the number of word images falling into each sortation category. The next 9 columns report the percentage of words falling in the top 9 ranks. The second to last column reports the percentage of words which were segmented into the correct number of characters (i.e. the correct string was not rejected) but which ranked at level 10 or greater. The final column reports the percentage of words for which the correct string was rejected due to incorrect character segmentation.

Table 2.6.4 - City Test Results

		Percent Correct in Top										
BEZ	cnt	1	2	3	4	5	6	7	8	9	>=10	Reject
9	220	85.5	85.9	87.3	87.7	87.7	88.6	88.6	88.6	88.6	88.6	11.4
5	340	78.8	80.0	80.3	80.3	80.3	80.9	80.9	81.2	81.2	81.2	18.8
Rej	274	60.9	64.6	65.7	65.7	66.1	66.4	66.4	66.4	66.8	70.1	29.9
All	834	74.7	76.5	77.3	77.5	77.6	78.2	78.2	78.3	78.4	79.5	20.5

Table 2.6.5 gives the city word verification test results when only words which were segmented in to the correct number of characters are considered. Note that segmentation into the correct number of characters does not imply completely correct character segmentation. In some cases the correct number of zones is found, but the zones do not line up correctly with the character boundaries.

The first column of table 2.6.5 gives the sortation category. The second column gives the number of words with correctly estimated word length (correct number of character zones detected). The third column gives this number of words as a percentage of the total number of word images present in the category (i.e. the cnt column in table 2.6.4). The next columns give the percentage of words falling in the given ranks. The final column gives the percentage of non-rejected words which will always be 100% since the table contains only words with correct length estimates.

Table 2.6.5 - City Results: Correct Word Length

			Percent Correct in Top									
BEZ	cnt	%	1	2	3	4	5	6	7	8	9	>=10
9	195	88.6	96.4	96.9	98.5	99.0	99.0	100.0	100.0	100.0	100.0	100.0
5	276	81.2	97.1	98.6	98.9	98.9	98.9	99.6	99.6	100.0	100.0	100.0
Rej	192	70.1	87.0	92.2	93.8	93.8	94.3	94.8	94.8	94.8	95.3	100.0
All	663	79.5	94.0	96.2	97.3	97.4	97.6	98.3	98.3	98.5	98.6	100.0

Table 2.6.6 gives the results of the street word verification test. The format of this table is identical to that of table 2.6.4.

Table 2.6.6 - Street Test Results

		Percent Correct in Top										
BEZ	cnt	1	2	3	4	5	6	7	8	9	>=10	Reject
9	220	86.2	86.2	86.2	86.6	86.6	86.6	86.6	86.6	86.6	87.0	13.0
5	340	83.5	84.4	84.4	84.4	84.4	84.4	84.7	84.7	84.7	85.3	14.7
Rej	274	78.1	80.4	81.1	81.5	81.9	81.9	81.9	81.9	81.9	83.3	16.7
All	834	82.6	83.6	83.9	84.1	84.2	84.2	84.4	84.4	84.4	85.2	14.8

Table 2.6.7 gives the street results for words with correctly estimated word lengths. The format of the table is identical to that of table 2.6.7.

Table 2.6.7 - Street Results: Correct Word Length

			Percent Correct in Top								
BEZ	cnt	%	1	2	3	4	5	6	7	8	9	>=10
9	215	87.0	99.1	99.1	99.1	99.5	99.5	99.5	99.5	99.5	99.5	100.0
5	284	85.3	97.9	98.9	98.9	98.9	98.9	98.9	99.3	99.3	99.3	100.0
Rej	225	83.3	93.8	96.4	97.3	97.8	98.2	98.2	98.2	98.2	98.2	100.0
All	724	85.2	97.0	98.2	98.5	98.8	98.9	98.9	99.0	99.0	99.0	100.0

From these tables, it is apparent that the word verification module works well when the correct number of characters is returned by the character segmentation module. The integration of multiple segmentation hypotheses with the word verification module is very straightforward, and should allow significant increases in performance. Continued work on feature generation method may also provide some improvement in performance.

2.7 Model Based Parsing

Once candidate line and word images have been segmented from the input address block image, it becomes necessary to identify roles (e.g. suffix word, street number, etc.) that specific word images play in the interpretation of the mailpiece. By identifying a small number of word roles in the image, it becomes possible to identify other words through the inherent structure of the address block. For example, if there exists a three word line and the first and third words are identified as a street number field and a suffix field respectively, then the second word can be classified as a street word field with high probability.

Although seemingly simple in theory, this process becomes extremely difficult in practice due to such limitations as image quality, segmentation failures, word verification, and patron errors. For these reasons, a hierarchical model-based approach was adopted that arrived at multiple interpretations for an input address block image. These interpretations are ranked and the overall system control decides how many are propagated forward for further system processing.

The heart of the parsing subsystem is a set of hierarchical models describing the structure of an address block image. The top level of the hierarchy is occupied by the model AB1 as shown below in Figure 2.7.1. This model seeks to interpret the address block image from a line level. In other words, the line(s) that contain the city, state, and ZIP Code fields are identified, followed by the line(s) that contain the primary name information (i.e. street name or post office box).

```
(defvar ab1    (:opt (:multifield 2 :dummy-line))
               (:req city-state-zip-line)
               (:opt (:match-line secondary-line))
               (:req primary-line)
               (:opt (:match-line secondary-line))
               (:opt (:multifield 4 :dummy-line))
               (:and-nothing-else)))
```

Figure 2.7.1 Top Level Address block Model

A model is composed of a list of clauses that are applied to the lines in the image from the bottom to the top. The first entry in each clause is the clause type. Four clause types are visible in Figure 2.7.1: :opt, :multifield, :req, and :match-line.

The :opt clause type refers to fields that might be present in the image, but do not necessarily have to be present. The first line field that needs to be identified in the image is the city-state-zip line. Although this is usually the last line in an address block, it does not necessarily have to be. Therefore, the first clause in AB1 allows for up to 2 dummy lines to be present at the bottom of the image. The multifield clause type is simply used to indicate that more than one line can occupy the title of :dummy-line. Figure 2.7.2 shows the state of parsing after the processing of this clause for a 3-line address block image.

Three parses are generated from the processing of this clause. The first parse occurs from ignoring the optional nature of the clause. In this parse, no line images are assigned to the :dummy-line variable and three lines remain to be interpreted. In the second parse, one line image is assigned to the :dummy-line variable and two lines remain for interpretation. Similarly, the last parse occurs when two lines are assigned to the :dummy-line variable and one line remains for interpretation.

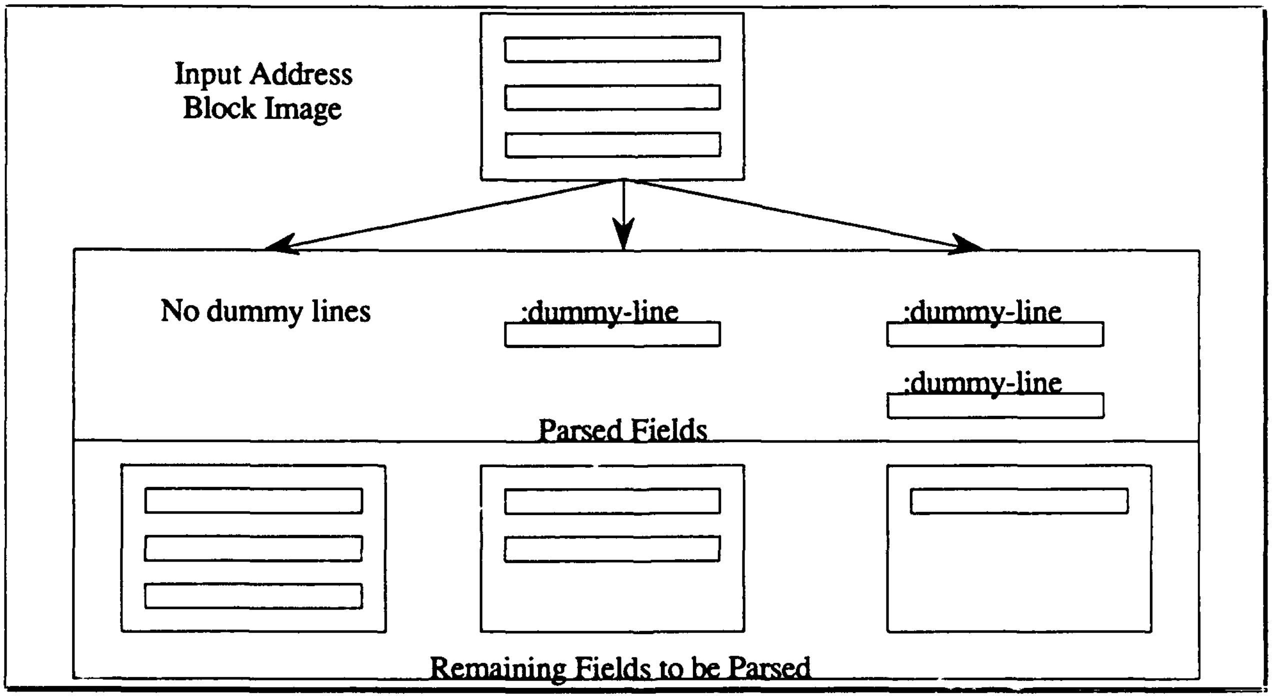


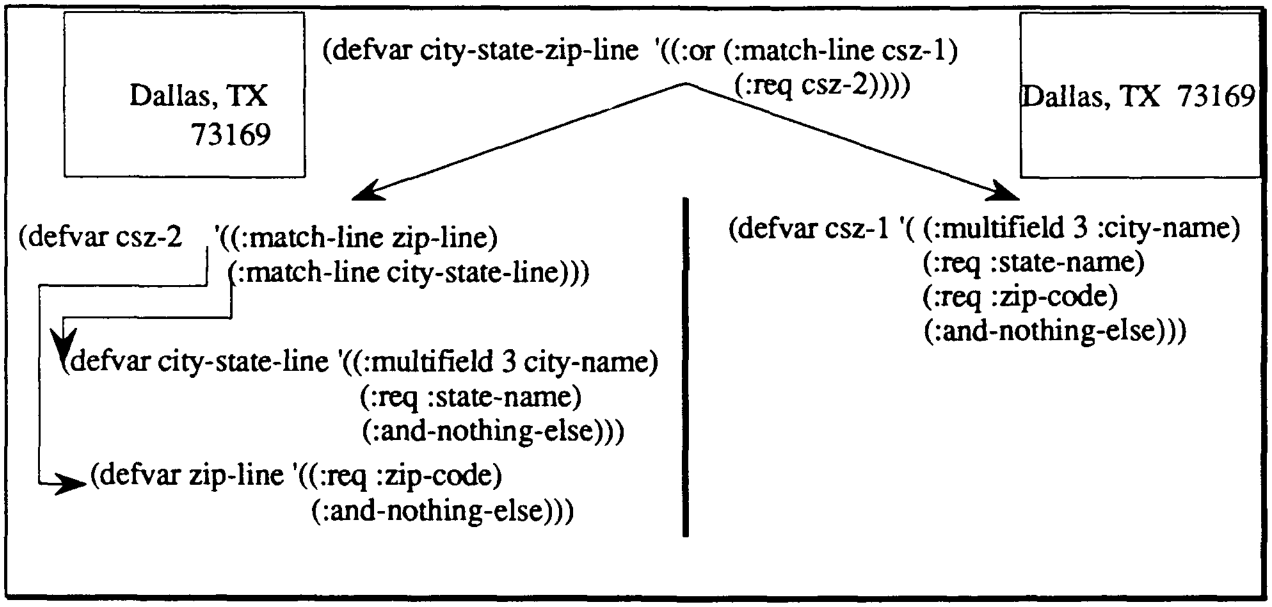
Figure 2.7.2 Top Level Parse Example

As the name implies, the dummy-line variable is used only as a placeholder for lines that do not contain interesting word fields within them. Therefore, at this stage in the parsing, all three of these candidate parses would be considered equally probable. The next line in the AB1 model is a

:req clause that is used for locating the city, state, and ZIP Code fields in the image. Notice that the "city-state-zip-line" word in the model is not preceded by a colon as was the case with the :dummy-line word. This syntax indicates a call to a sub-model. The city-state-zip-line sub-model is shown below in Figure 2.7.3.

The modelling language currently in place in the Phase II system supports two formats for city, state, and ZIP Code field placements which is also shown in Figure 2.7.3. The first format, shown in the left half of the figure, is for the city and state fields to be placed on one line with the ZIP Code field located by itself on the line directly below the city and state line. The second format is the most common and consists of the city, state, and ZIP Code fields to all be located on one line. The top level city-state-zip-line model is at the top of this figure and implements these two formats through its :or clause.

The :match-line clause, which is contained in the city-state-zip-line model, is the method in which the hierarchical nature of the parsing is implemented. All of the matching discussed thus far has dealt with associating line images to variable names. The :match-line clause causes the parsing to match the word images contained in the next available line image to word variables contained in a line model (in this case, the csz-1 line model). All of the variable names contained in the csz-1 model (:city-name, :state-name, and :zip-code) refer to word images.

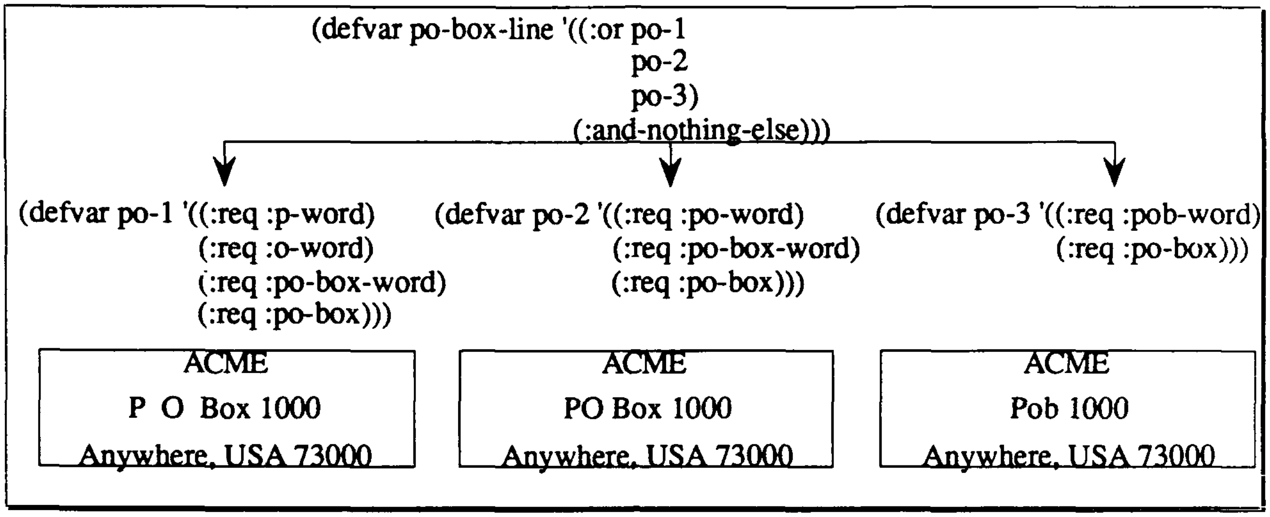


```
(defvar city-state-zip-line '((:or (:match-line csz-1)
                                   (:req csz-2))))

(defvar csz-2 '((:match-line zip-line)
                (:match-line city-state-line)))

(defvar city-state-line '((:multifield 3 city-name)
                          (:req :state-name)
                          (:and-nothing-else)))

(defvar zip-line '((:req :zip-code)
                   (:and-nothing-else)))

(defvar csz-1 '( (:multifield 3 :city-name)
                 (:req :state-name)
                 (:req :zip-code)
                 (:and-nothing-else)))
```

Figure 2.7.3 City-state-ZIP line Models

Similar to the structure of the city, state, and ZIP Code assignments, the primary-name submodel, called from the AB1 model, is a combination of submodels that describe the numerous formats in which the more specific destinating address information can be arranged. It consists of two main submodels, each of which describes the two main forms of primary name information: post office box lines and street name lines.

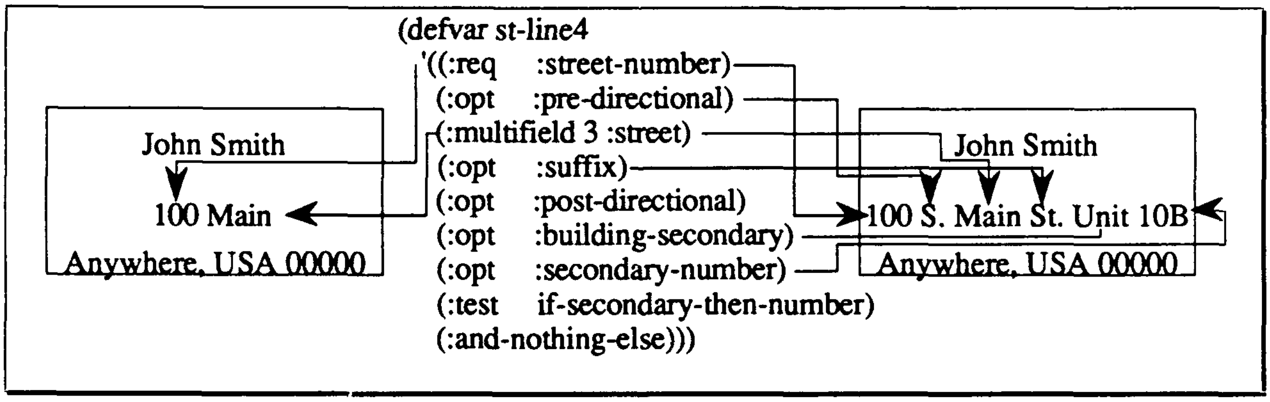
The post office box model itself consists of three different submodels each describing a particular spatial arrangement of words that could constitute a post office box line. These three models and examples are shown in Figure 2.7.4. Nuances in word segmentation dictate the need for three submodels because the key words of post office box lines (e.g. "P", "O" "PO") are occasionally segmented into one word and occasionally two words.



```
                    (defvar po-box-line '((:or po-1
                                               po-2
                                               po-3)
                                          (:and-nothing-else)))

(defvar po-1 '((:req :p-word)      (defvar po-2 '((:req :po-word)       (defvar po-3 '((:req :pob-word)
               (:req :o-word)                     (:req :po-box-word)                  (:req :po-box)))
               (:req :po-box-word)                (:req :po-box)))
               (:req :po-box)))
```

Figure 2.7.4 Post Office Box Submodels

The second component of the primary name model is the st-line4 submodel. This model is shown in Figure 2.7.5. Along with the submodel, this figure illustrates the two extreme examples of street line models. The simplest form of a street line is on the left and consists of just a street number and a one word street name. An examination of the st-line4 model shows that these two variables, :street-number and :street, are the only two variables which are required (A :multifield clause requires at least one field to be matched). However, this model also has numerous :opt clauses whose use is demonstrated on the more cluttered example on the right. This example contains not only a street number and street, but also contains a pre-directional ("S"), a suffix ("St"), a building secondary modifier ("Unit"), and a building secondary number ("10B"). One of the strengths of the modelling language and the parsing subsystem as a whole is that allows a very concise representation to model both of these extremes and any interpretation in between.



```
(defvar st-line4
 '((:req   :street-number)
   (:opt   :pre-directional)
   (:multifield 3 :street)
   (:opt   :suffix)
   (:opt   :post-directional)
   (:opt   :building-secondary)
   (:opt   :secondary-number)
   (:test  if-secondary-then-number)
   (:and-nothing-else)))
```

Figure 2.7.5 Street Line Model

Up until now, the discussion has centered on the ability of the parsing subsystem to generate multiple interpretations for an input address block image. However, the topic of keeping these interpretations ranked amongst themselves has been carefully avoided. This task is accomplished primarily through the use of word verification on selected word images.

Certain word roles, such as suffix and pre-directional, have a fairly limited lexicon of possible strings. For example, the strings "North", "South", "East", "West", and their associated abbreviations cover the possibilities for the pre-directional field. Since this lexicon is limited in size, it becomes computationally feasible to verify the entire lexicon against a potential pre-directional word image. The corresponding confidence that the candidate word image does indeed play the role of pre-directional is the confidence of the highest matched string in the lexicon. This procedure has the additional benefit of identifying the specific pre-directional string present.

This overall method is used for the detection and identification of state words, pre-directionals, suffixes, post-directionals, building secondary identifiers, and post office boxes. Table 2.7.1 shows the lexicons used for each of these roles. These words are matched in upper case and mixed case. In addition, abbreviations of the words in the table are used and punctuation is also added.

Table 2.7.1 Word Role Field Lexicons

State Words	Texas
Pre-directionals	North, South, East, West
Suffixes	Lane, Road, Drive, Trail, Court, Street, Avenue, Circle, Freeway, Parkway, Expressway
Post-directionals	North, South, East, West
Bldg. Sec. Identifier	Floor, Number, Lock Box, Apartment, Suite, Mail, Box, Station
Post Office Boxes	P, O, PO, Pob, Post, Office, Box

Numeric word role fields, such as street number, ZIP Code, building secondary number, and post office box number, are identified through a procedure that attempts to separate numeric word images from non-numeric word images. A candidate word image is first segmented into characters. At each character position, all numeric characters (0-9) are verified and the character with the highest confidence is assigned to be the most likely numeric character at that position. This is repeated for every character position and the character confidences are summed to arrive at a confidence that the word is a numeric word. This process on the word is repeated for upper case characters (A-Z) and mixed case (A-Z first character, a-z other characters). The formula below is used to compute the overall confidence that the word image contains a numeric field.

$$\text{Number Confidence} = \frac{C_N}{C_U + C_M} \text{ where}$$

C_N = confidence using most likely numeric characters
C_U = confidence using most likely upper case characters
C_M = confidence using most likely mixed case characters

A final confidence assignment is that of street word. Most of the confidence assignments discussed thus far use word verification with a limited lexicon size. Due to the prohibitively large lexicon associated with street words, this approach is not feasible. Instead a technique is used that uses negative information derived from the word image for other word role fields. In practice, the following formula is used.

$$\text{Street Confidence} = 1 - \text{Max}(C_D, C_S, C_B) \text{ where}$$

C_D = Directional Confidence (Pre or Post)
C_S = Suffix Confidence
C_B = Building Secondary Identifier Confidence

Figure 2.7.6 shows some typical address block word images and their associated confidences for specific word roles. Several interesting features exist in this figure. Firstly, the "None" entries in this figure indicate fields where the word verification subsystem did not attempt to match any word in the respective lexicon to the input word. For example, the image of the "W" pre-directional word has a "None" entry in its suffix column. This is due to the fact that the word image was correctly segmented into one character and there are no one letter strings in the suffix lexicon.

A second interesting feature of this figure arises in the row for the "323" word image. This word image actually plays the role of the street number and its corresponding number confidence is 0.683. However, it has a confidence of playing a street word of 0.9998. This illustrates another strength of the model-based approach in that the models impose a structural constraint upon the address block. In other words, this word image has a strong indicator of being a street word, but it is also the first word in the line and therefore cannot play the role of a street word due to the structure imposed through the syntax of the models.

A final note regarding this figure deals with the pre-directional "W" row. Under the directional column, it is seen that the first choice for the pre-directional string is "N" instead of "W". This is caused by the difficulty associated with verifying word images of short length and illustrates the need for a parsing system that propagates multiple hypotheses forward through the end-to-end system.

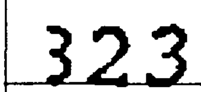
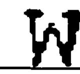
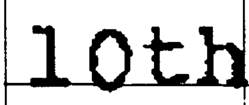
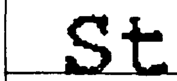
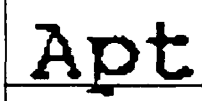
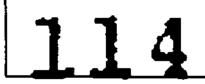
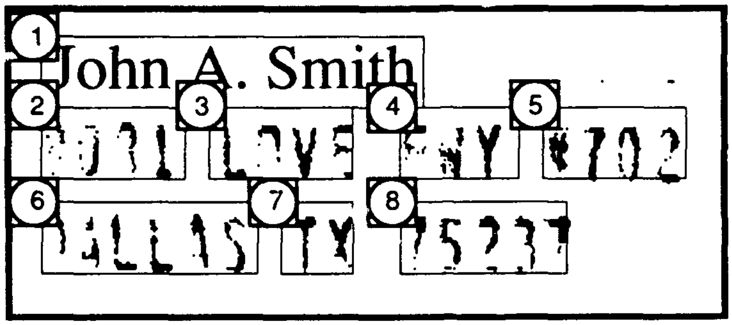
Image	Number	Directional	Street	Suffix	Building Secondary
	0.683	None	0.9998	"AVE" 1.21e-6	"BOX" 1.66e-4
	0.194	"N" 0.9 "W" 0.8	0.100	None	None
	0.263	"W" 5.4e-10 "E" 2.8e-10	0.9999	"ROAD" 2.9e-5	"Apt." 5.4e-7
	0.009	"S." 2.21e-5	0.374	"ST" 0.626	"FL" 1.6e-7
	0.114	None	0.0076	"AVE" 0.009	"Apt" 0.992
	0.351	None	0.982	"CIR" 0.018	"STE" 1.32e-4

Figure 2.7.6 Word Role Examples

An example of the parsing output is given in Figure 2.7.7. The top half of this figure contains a sample address block image with the word images segmented into numbered blocks. The bottom portion of the figure contains the top 2 parses that are generated for this image. The two parses are very similar in nature with the city, state, zip code, street number, and top lines all being identified correctly. However, the entries in bold indicate where these two parses differ. The top ranked parse correctly identifies image 3 as the street word, image 4 as a suffix word, and image 5 as a secondary number. It is interesting to note that the suffix-id field contains more than one entry ("FWY" with a 0.91 confidence and "HWY" with a 0.77 confidence). The overall parse confidence is generated using the confidence of the most likely suffix word (i.e. 0.91), but alternate values are allowed for certain word roles and are taken advantage of in the verification processing that follows address block parsing.

The second parse incorrectly identifies the "FWY" image as a street word and the "#702" word as a suffix word of "ROAD". Since the verification of "ROAD" resulted in a 0.42 confidence, this parse as a whole is ranked behind the top parse.



Parse 1		**Parse 2**	
:CITY-NAME	(6)	:CITY-NAME	(6)
:TX-WORD	7	:TX-WORD	7
:TX-WORD-ID	"TX"	:TX-WORD-ID	"TX"
:TX-WORD-CONF	0.75	:TX-WORD-CONF	0.75
:ZIP-CODE	8	:ZIP-CODE	8
:ZIP-CODE-CONF	0.42	:ZIP-CODE-CONF	0.42
:STREET-NUMBER	2	:STREET-NUMBER	2
:STREET	(3)	**:STREET**	(3 4)
:SUFFIX	**4**	**:SUFFIX**	**5**
:SUFFIX-ID	**(("FWY" 0.91)**	**:SUFFIX-ID**	**(("ROAD" 0.42))**
	("HWY" 0.77))	**:SUFFIX-CONF**	**0.42**
:SUFFIX-CONF	**0.91**	:DUMMY-LINE	(1)
:SECONDARY-NUMBER	**5**	Parse Confidence	0.00156
:DUMMY-LINE	(1)		
Parse Confidence	0.0034		

Figure 2.7.7 Parsing Example

Much work remains on the parser for Phase III. An examination of the ECA supplied truth files show that the models only cover approximately 80% of our image database. The modelling language will have to be expanded to account for more of the sample space with one of the more notable improvements being the addition of Rural route parsing.

The parsing subsystem also only contains the correct parse as a member of its top three parses approximately 50% of the time. While much of this can be explained by the various image segmentation subsystems (line, word, and character), word verification, and image quality, a significant portion remains that is due to various attributes of the parsing subsystems. Avenues of investigation for Phase III will include other methods of confidence assignment and the intelligent incorporation of prior word role location probabilities.

2.8 Online Databases

The contextual analysis system depends heavily on a database derived from the ZIP+4 and City/State database supplied by the USPS. This section describes the design of the contextual analysis system's database, and discusses issues surrounding its construction. The use of the database during system operation is described in the section on system control entitled "Query Processing and Primary Lookup".

The configuration of the current database design is given in Figure 2.8.1. This structure breaks the database down into two files: a primary name file and an add-on file. The fields present in each file are shown in the figure. The ZIP+4 records which share primary components are compressed into one record in the primary name file. The various secondary components, secondary name, #, and add-on numbers, are referenced from the primary name file by pointers into the add-on file.

In addition to the two database files, the database structure includes two hash tables which allow quick lookup of certain groups of records from the target address files.

The main look process is accomplished through the use of the query dictionary hash table. This hash table supports 10 kinds of database queries as shown in table 2.8.1.

Table 2.8.1 - Main Hash Table Query Types

Query Type	Argument Type	Matches If Argument Equals
now-pri	number	number of words in the primary name field
wl-pri-N	number	length of Nth word in primary name field (N = 1,2,...,9)
mc-pri	string	contents of the primary name field
mc-suf	string	contents of the suffix field
mc-pre	string	contents of the pre-directional field
mc-pos	string	contents of the post-directional field
mc-zip	string	contents of the ZIP code (5 digits) field
mc-cit	string	contents of city name field

Each query has a type and an argument. The result of the query is the list of target record numbers which satisfy the query. For example, the response to the query (mc-pri "NORTH DALLAS") would be the list of record numbers for all target records whose primary name field contained the string "NORTH DALLAS". From this list, the actual records for "NORTH DALLAS" can be retrieved.

A second hash table, the city query hash table, has a slightly different function. Table 2.8.2 describes the queries handled by this hash table. The response to queries in this hash table is a list of all city names which satisfy the given query. These strings can then be used in the mc-cit query for the main query dictionary to retrieve record numbers for those city names.

Table 2.8.2 - City Hash Table Query Types

Query Type	Argument Type	Matches If Argument Equals
now-cit	number	number of words in the city name field
wl-cit-N	number	length of Nth word in city name field (N = 1,2,...,9)

The final element of the overall database structure is the primary name dictionary. This is a hash table with keys being target record numbers, and output being the primary name for that record number. This information may also be retrieved from the primary field in the target address database, but the in-core hash table speeds this operation.

Several issues discovered in Phase I motivated the redesign of our inverted target address database structure. First, the control of our end to end system deals mainly with the verification of primary names with the primary name image contained in the candidate address block image. Each query that was processed by our Phase I system had to perform repeated file access in order to retrieve entire address records just to extract their primary names. This was an extremely costly and inefficient process.

Secondly, it was discovered that there existed a substantial amount of repetition in the primary name field of our database files. This meant that unnecessary target record lookups were performed in order to retrieve primary names previously extracted. It also meant that our database files were larger than necessary.

The Phase I database system was also a two file configuration with primary and add-on files. However, in the Phase I system, target records were repeated in the primary name file if they differed in the predirectional, suffix, postdirectional, secondary name, or record type fields. In the new design, primary name entries are only repeated in the primary name file if they differ in the predirectional, suffix, or postdirectional fields.

The ZIP+4 files for the four SCF's in question contain 298,678 records. In its old configuration, the primary name file consisted of 62,857 records, yielding a compression rate of 4.75:1. In the new configuration, the primary name file contains 40,029 records, or a compression rate of 7.46:1.

An even more attractive feature of the new structure is the incorporation of the primary name hash table. Queries to the query dictionary return lists of target numbers. These target numbers are indices into the primary name file to be used to retrieve entire address records. In addition, they are used as keys into a primary name hash table for efficient retrieval of the primary name component of the address records. This greatly increases system throughput.

The third and final component of the database is the city query hash table, also shown in figure 2.8.1. The city query hash table is used during dynamic city name verification. Keys into this table are word length and number of word queries which come from the address block parse. The output is a list of cities in the four SCFs which match the queries. These city names are verified and the top choices are used as match-contents queries during dynamic primary verification.

This work has also exposed two flaws with the ZIP+4/City-State file structure. The first concerns the merging of the ZIP+4 database with the City-State database. The recommended method of attaching city names to target records from the ZIP+4 is to extract the 5 digit ZIP Code and the 6 digit finance number from each target record and assign the city name from the City-State file possessing the same ZIP/Finance number pair. However, upon implementation of this technique, we found that a specific target record could result in multiple city names when merging with the City State file. This occurs for non-unique ZIP Codes, record type 80 in the City-State file. Table 2.8.3 gives a tabular picture of the severity of the problem. Each entry in this table lists the problem ZIP Code and finance number, the conflicting city names, and the number of target records having this conflict. The problem can be summarized by noting that a total of 60,142 target records have a multiple city name conflict. This number reflects almost 20% of the total number of target records.

In the database, in situations where there are multiple cities, the primary record is repeated, with a change in city name. The resulting increase in number of primary records causes some inefficiency in the system, but this appears to be the cheapest solution. As a result of the multiple-city problem, the number of records in the primary component increases from 29,158 to 40,029. This number is still considerably lower than the 62,857 primary records in the old database. Also, the increase in size of primary component does not affect the size of the add-on component, since records which correspond to multiple cities point to the same add-on records.

The second problem with the ZIP+4 database deals with incomplete and ambiguous records. We classify an incomplete target record as one without 5 digits in its ZIP Code or less than 4 digits in its add-on low or add-on high field. 2,368 records in our ZIP+4 files are classified as incomplete. Incomplete records are not inserted in our revised database structure.

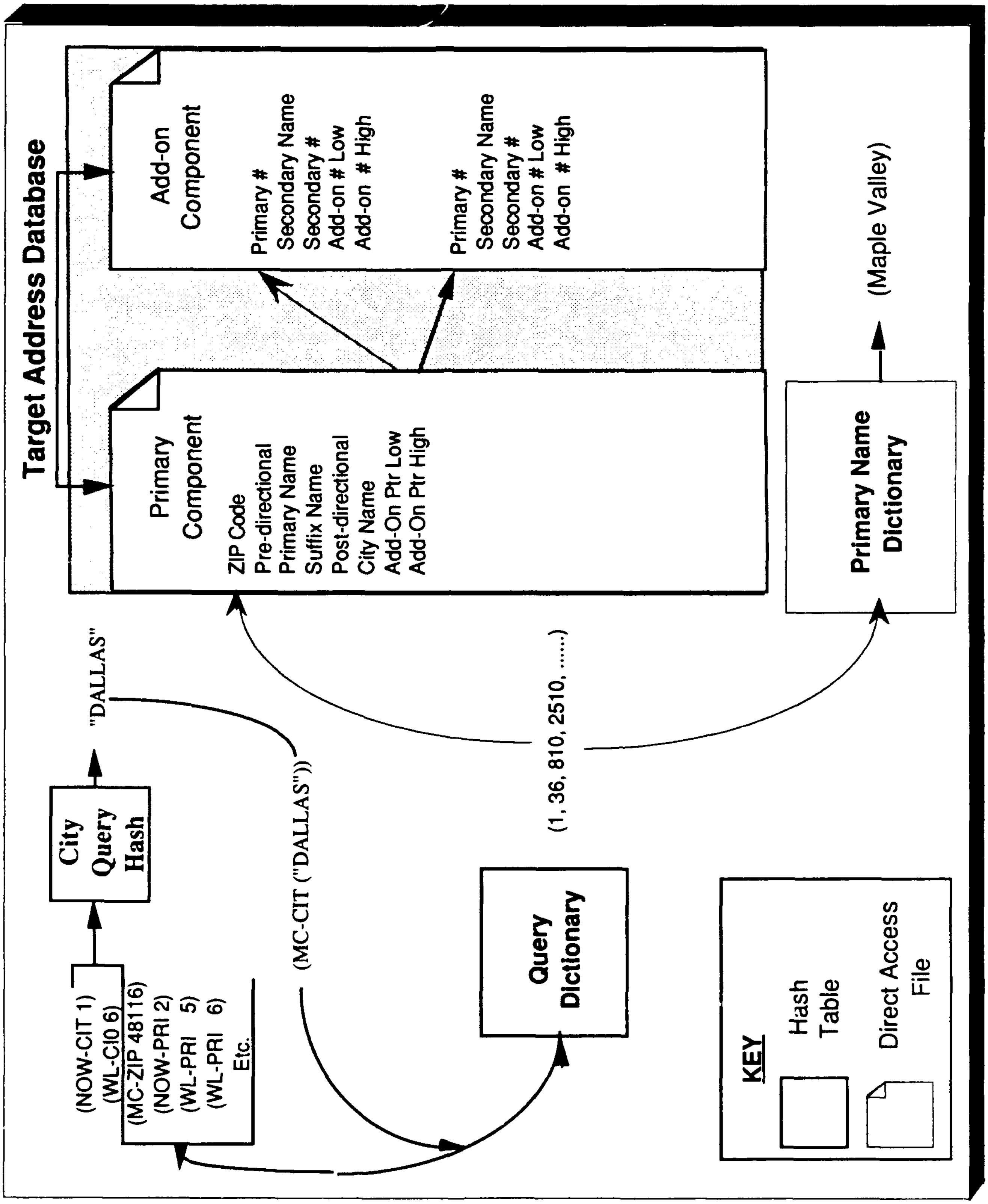


Figure 2.8.1 New Database Structure

Table 2.8.3 City-State/ZIP+4 merging problem

Zip Code	Finance #	Cities	# Records
75059	484150	Howe/Dorchester	88
75074	487110	Plano/Murphy	3158
75147	485525	Mabank/Gun Barrel City	1767
75119	482905	Ennis/Telico	2987
75115	482420	De Soto/Desoto/Glenn Heights	3410
75040	483410	Garland/Sachse	4195
75069	485490	Mc Kinney/McKinney/Lucas/Fairview	4468
75244	482270	Dallas/Farmers Branch	2537
75182	485860	Mesquite/Sunnyvale	234
75151	482040	Corsicana/Navarro	1
75234	482270	Dallas/Farmers Branch	2711
75028	485130	Lewisville/Flowermound/Flower Mound	2705
75076	487255	Pottsboro/Fink	412
75110	482040	Corsicana/Navarro/Emhouse	4700
75067	485130	Lewisville/Highland Village/Double Oak	4503
75038	484360	Irving/Las Colinas	3853
75087	487710	Rockwall/Heath	2642
75211	482270	Dallas/Cockrell Hill	4358
75080	487555	Richardson/Buckingham	3162
75205	482270	Dallas/Village/University Park/ Park Cities/Highland Park	3169
75039	484360	Irving/Las Colinas	501
75253	482270	Dallas/Kleberg	1149
75056	485130	Lewisville/The Colony	1030
75143	484625	Kemp/Seven Points	535
75180	485860	Mesquite/Balch Springs	1867

Ambiguous target records are records that a) possesses the same primary component information, b) agree in the primary number odd/even field, c) overlap in their primary number range field, and d) have primary number ranges that are greater than one. 169 target records in the ZIP+4 are classified as ambiguous. An additional 48 were classified as redundant, differing only in the action code and/or update key fields.

2.9 Query Processing and Primary Lookup

The query processing and primary lookup module of the end-to-end system accepts as input the top seven parses generated by the model-based parsing routines. It uses these parses to generate a set of queries each of which is constructed from a specific parse. The basic subcomponents are shown in figure 2.9.1

Once an address block image is parsed, certain pieces of information within the parse can be extracted to focus a search for potential target records in the ZIP+4 database. If the top-ranked parse is a Post Office Box parse, processing proceeds in a special manner which will be described later in detail. Otherwise control is handed over to the query initialization module. In this module, an exact query is constructed from each input parse by extracting the maximum amount of querying information from each parse.

Next the queries resulting from these parses are input to a query dictionary that returns a list of target numbers for each piece of information contained in each given query. These lists are then intersected yielding one list of target numbers for the entire query. The queries are then ranked, based on the confidence of the query, the confidence of the parse it came from, and the number of targets contained in the intersected target list. A winner is chosen.

Next, control is assumed by the primary lookup and verification module, which retrieves the primary names corresponding to each element of the winning query's target list by keying into the primary name hash table with individual target numbers. Each primary name is then verified. If the maximum verification value exceeds a threshold, the system moves on to add-on processing. Otherwise, query backoff occurs.

Query backoff postulates that, due to incorrect parsing or patron errors, some of the information in the query may have been incorrect. To compensate for this, new queries are generated by dropping pieces of information out of the exact query, or using alternate values for certain parse fields. The resulting queries are then re-ranked to ensure that the more confident queries are processed first the next time around. This cycle proceeds until a high verification value is returned by the primary verification module, or until one of several timeout criteria are met. Each of the components of query processing and primary lookup are covered in detail below.

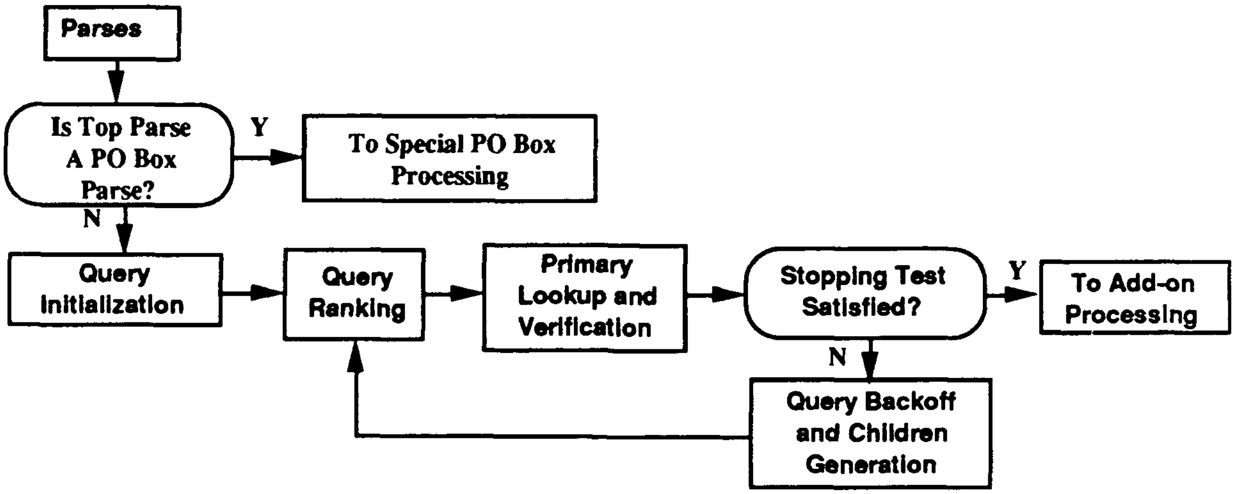

Parses
Is Top Parse A PO Box Parse?
Y
To Special PO Box Processing
N
Query Initialization
Query Ranking
Primary Lookup and Verification
Stopping Test Satisfied?
Y
To Add-on Processing
N
Query Backoff and Children Generation

Figure 2.9.1 System Control Overview

The query initialization module accepts the parses generated by the address block parser and constructs a set of exact queries for each parse. Since the parses are ranked, exact queries :re constructed only for the top 7 parses. These exact queries are constructed by extracting certain pieces of information from the parse. Currently, four kinds of queries are generated as a result of this action. First, Match-Contents queries are generated for pre-directionals, post-directionals, and suffixes. Second, Number-of-Word and Word-Length queries are generated for the primary name. Third, a Match-Contents query is generated for the ZIP code using the MLOCR result for the mailpiece. Fourth, a Match-Contents query is generated for the city by using dynamic city verification, which is described below. Exact query initialization pulls all of these categories from a given parse if they are present, and constructs the most focussed query possible for the respective parse.

The goal of dynamic city verification is to identify the city name on the mailpiece. Dynamic city verification proceeds by first generating word length and number of word queries for the parsed city fields (see figure 2.9.2 below). These queries act as keys into a city query hash table, whose output is a list of all possible cities which match the inputted queries. Each of the candidate city names is verified. If none are verified with a confidence greater than 0.9, the system backs off of one or more queries and tries again. Also, the system is allowed to back off of one or more of the image fields parsed as a city field. This type of back off postulates that the field is either image noise or is incorrectly parsed. Finally, it should be noted that each city name is also verified with a comma appended at the right end of the final city word. Ultimately, the top candidate city name is used to generate a match-contents query for primary lookup and verification.

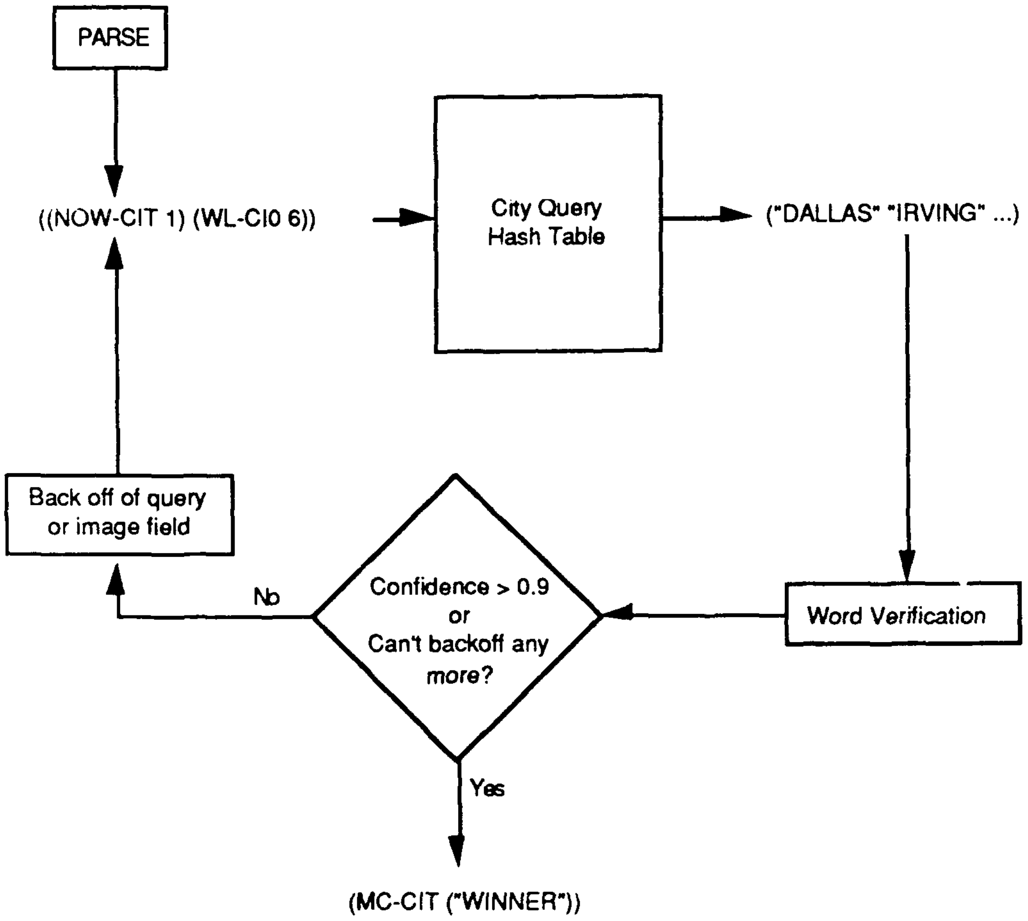


Figure 2.9.2 Dynamic City Verification

Once the queries have been created, the list of target numbers that satisfy each query must be calculated. Very early on in the project, the ZIP+4 database was merged with the City-State database in a compressed manner to form the Target address database. Each entry in this database is called a target record and corresponds to a particular set of addresses. A target number has a one-to-one correspondence with a specific target record in this database. Since the database was constructed as a fixed record length direct access file, it is fairly quick to look up a given target record when given its target number.

The concepts of target records and target numbers were introduced as a means of increasing the focusing speed of the system. The individual pieces of information that compose a potential query (i.e. (mc-suf) (mc-ZIP), etc.) were precomputed on the target address database. The result of a precomputation of an individual piece of information, say (mc-ZIP 48116)), is a list of target numbers whose target records are located in the 48116 ZIP code area. These precomputed results were stored in a component called the query dictionary. The structure of this component is a hash table, so the precomputed results are quickly and efficiently retrieved by the system.

The query dictionary is used to generate the list of target numbers that satisfy a given query. Each individual piece of information contained in the query is queried into the dictionary to find the target numbers that satisfy that piece. Then the results for all of the individual pieces of information are intersected to find the list of numbers that satisfy all of the pieces of information and, hence, the overall query.

The subcomponent that intersects all of the target lists from the individual pieces of information is appropriately called the ordered intersection module. In order to efficiently perform its operation, it leverages off of the fact that each query into the query dictionary returns a list of numbers that are strictly monotonically increasing. This simplification is extremely important due to the sheer volume of queries and individual pieces of query information.

When the target lists corresponding to each query have been created, the target records for all target numbers must be retrieved from the target record database so their primary names can be verified against the primary name contained in the address block image. Again, due to the volume of potential queries and, more importantly, target numbers, it is desirable to order the queries and target lists according to some measure of the probability of the correct target record being contained in a query's target number list. Therefore, at this point, processing enters a Query Ranking phase.

The input to the query ranking component is a set of potential queries, their confidences, the target number lists associated with each individual query, ar.d the parses that each query originated from. It is desired that the target number lists be ranked by some function that accounts for the query probability, the parse probability, and some measure of the work involved to process that given target number list. The formula currently in place is:

$$R_{TNL} = \frac{C_Q}{\max(20, L_{TNL})} * C_P$$

where L_{TNL} is the length of the target number list, C_Q is the confidence of the potential query, C_P is the confidence of the parse that the query was generated from, and R_{TNL} is the resulting rank of the target number list. This is a heuristic function that uses the potential query's confidence to give weight to more probable queries. The length of the target list is used as an estimate of the work required to further process the potential query. It assumes a worst case scenario in which every target number in the list corresponds to a target record with a unique primary name. In this case, the length of the target number list equals the number of word verifications that must be performed in the primary verification system component. There is a substantial amount of overhead necessary to initially perform word verification on a candidate string. Eventually, however, the system arrives at a point where performing verification on another candidate string is not very computationally intensive. It is for this reason that the target number list length is compared against 20 in the denominator. Finally, the last factor, parse confidence, helps shift emphasis to queries that originated from higher ranked parses.

The winning query is the query with the highest confidence value according to the formula given above. The Target Address Lookup module retrieves the primary names corresponding to each target number on the query's target number list. This is done using the primary name hash table, which has a one-to-one correspondence to the primary component of the primary component of the database. The primary names are then used in the Primary Verification process which is described in the next section. After all of the primary names have been verified, a decision is made whether to back off and consider more queries or to move on to add-on processing. Currently if any one of four ending conditions is met, the system moves on to add-on processing. Those four ending tests are:

1. Verification Value > 0.9
2. Total Number of Verifications >= 750
3. No More Queries to Back Off Of
4. More than 7.5 Minutes Elapsed Since Start of
   Query Processing and Primary Lookup

If none of the stopping conditions is met, then query backoff occurs. There are two kinds of query backoff - omission and replacement. In query omission, a single query is left out of the aggregate query. This results in a more general query of the database. In query replacement, a single query is changed to an alternate value. For an example of query backoff, see figure 2.9.3 below. The initial aggregate query contains four pieces of information, and there are two alternate pieces of information from the parse which are not currently being used. By leaving out a single query, four new and more general queries are made. By replacing queries with the alternate pieces of information, two new and different queries which result from the same parse structure can be tried.

1. Active Aggregate Query -
   ((mc-pre "N") (mc-suf "DR") (mc-cit "DALLAS") (mc-zip "75240"))
2. Alternate Queries -
   (mc-pre "W") (mc-suf "LN")
3. Backoff by Omission -
   ((mc-pre "N") (mc-suf "DR") (mc-cit "DALLAS"))
   ((mc-pre "N") (mc-suf "DR") (mc-zip "75240"))
   ((mc-pre "N") (mc-cit "DALLAS") (mc-zip "75240"))
   ((mc-suf "DR") (mc-cit "DALLAS") (mc-zip "75240"))
4. Backoff by Replacement -
   ((mc-pre "W") (mc-suf "DR") (mc-cit "DALLAS") (mc-zip "75240"))
   ((mc-pre "N") (mc-suf "LN") (mc-cit "DALLAS") (mc-zip "75240"))

Figure 2.9.3 An Example of Query Backoff

From here, control would be passed back to query ranking, and another pass would be made at Primary Verification. This three-step cycle of Query Backoff, Query Ranking, and Primary Verification continues until one of the four stopping criteria are met.

If the top-ranked parse is a Post Office Box parse, then the query and verify phase proceeds differently. In this case, four query combinations are tried in a predefined order. They are shown in figure 2.9.4 below. If after any query, an add-on from a single ZIP code is found, then the system succeeds. If add-ons from two or more different ZIP codes are found, then the system fails. Otherwise, the next query is tried.

1. PO Box + City + MLOCR Zip
2. PO Box + MLOCR Zip (if not default)
3. PO Box + City
4. PO Box + MLOCR Zip (if default)

Figure 2.9.4. PO Box Querying

3 System Performance

This section describes the results of a test of ERIM's contextual analysis system at the end of Phase II of the project. At the end of Phase I, a prototype system was in place with each of the system components having been developed to a proof of concept level only. The Phase I testing period targeted four key components for more aggressive development during Phase II. These components (System Control, Word Verification, System Database, and Image Segmentation) became the focus for Phase II of the project.

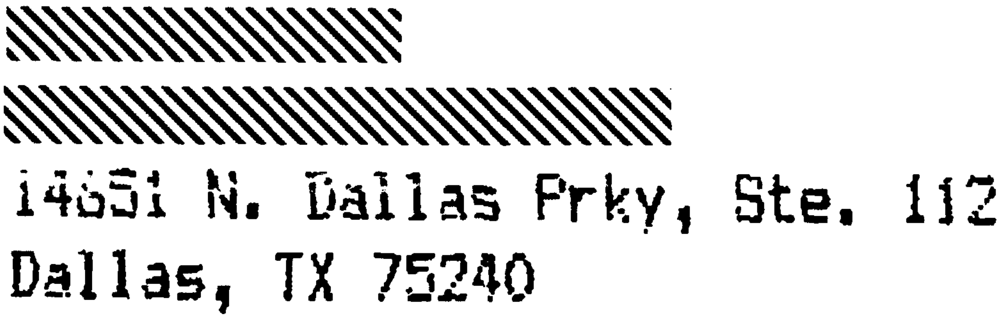
Our overall goal for Phase II was to complete construction of all new or improved subsystems two weeks prior to the Phase II testing period in order to thoroughly test our end-to-end system. All of the components met this deadline except the segmentation subsystem, and work will continue on this component in Phase III. As a result of this slight lag in schedule, little effort was directed at system level refinement before the testing period. The Phase II test was started, however, on time.

Because this test was initiated before a complete system checkout, the initial results of the Phase II test uncovered several control level refinements that were needed to the system. For this reason, this section provides additional system level results on untrained testing data.

As a stand alone system (i.e. the MLOCR BEZ response is not used when the CA system rejects the image), the Phase II results on 929 address block images can be summarized as follows: 319 (34.34%) of the images received a correct 9-digit ZIP Code, 43 (4.63%) of the images received an incorrect 9-digit ZIP Code, and 565 (60.95%) images were rejected by our system. During the Phase II testing period, several shortcomings of the system were identified and repaired. These repairs boosted system performance on the Phase II test images to 407 (43.81%) images processed correctly, 39 (4.20%) processed incorrectly, and 483 (51.99%) rejected. To test the robustness of these minor modifications, an independent test set of 959 previously unseen images was gathered from the training 1 dataset. The system again achieved a substantial level of performance by processing 403 (42.02%) images correctly, 37 (3.86%) incorrectly, and rejecting 519 (54.12%).

For the initial end-to-end system level testing, the 929 test images were completely processed by the end to end system. Due to an error in system processing, we were unable to recover results for two of the images, D00518DRM and D00554DRM. Therefore, all of the tables and percentages reported on the Phase II test images will be with respect to 927 total images. Both of the problem address block images were rejected by the end-to-end system.

When calculating the system performance, a mailpiece image is said to be processed correctly only if the 9-digit ZIP Code assigned to the mailpiece by the system corresponds exactly to one of the ZIP Codes included in the Level of Sortation (LOS) information for the image. In the past, we have usually found instances where there exists what we refer to as an LOS discrepancy. An LOS discrepancy applies when the system response for an image is classified as incorrect, but the target record extracted from the ZIP+4 database refers to the actual address on the mailpiece. In other words, the ZIP Code assigned to the mailpiece by the system does not match any of the LOS ZIP Codes, but is the correct ZIP according to the ZIP+4 database. An example of an LOS discrepancy (Image D00655CRM) is shown below in Figure 3.1.



[redacted]

[redacted]

14651 N. Dallas Prky, Ste. 112
Dallas, TX 75240

Figure 3.1 Sample LOS Discrepancy

Our processing of the address block image results in the assignment of a target record from the ZIP+4 database containing the following correct information:

ZIP Code	75240-8899
Primary Name	"Dallas"
Suffix	"Pky"
City Name	"Dallas"
Primary Number	14651
Secondary Name	Gallo Wine
Secondary Number Low	112
Secondary Number High	112

This information matches the input address block image exactly, even down to the secondary name of Gallo Wine. The LOS ZIP Codes supplied for this image are shown below:

"HA"	75240-7477
"H"	75240-7476
"S"	75240-7410
"5"	75240
"3"	75200

Unfortunately, the correct ZIP Code as we retrieved from the ZIP+4 target record (75240-8899) is not included in these "correct" ZIP Codes.

When analyzing the end-to-end system, post-processing of the results is employed to ensure that the LOS discrepancies are counted as correct. We will first present the results of the end-to-end test without this post-processing in order that a direct comparison can be made to the other contractors' results. All additional results, however, will include post-processing of LOS discrepancies in order to give an accurate assessment of our system.

3.1 Raw Phase II Test Set Results

The steps involved in the end-to-end processing were: 1) Hough transforming to remove tilt, 2) line detection and segmentation, 3) word segmentation, 4) model-based parsing of the complete image, 5) word length estimation, 6) database query generation and lookup, 7) image domain word verification and, 8) add-on lookup and verification. The results from this test, *without using the BEZ information*, can be summarized as follows: 290 (31.28%) of the images received a correct 9-digit ZIP Code, 72 (7.77%) of the images received an incorrect 9-digit ZIP Code, and 565 (60.95%) images were rejected by our system.

Five different tabular measurements were employed to further describe the system results. In all of these analyses, *we used the BEZ output when our system rejected the mailpiece.* The first table, as shown in Table 3.1-1, uses the depth of sortation to compare the performances of the LOS information, the BEZ output, and our performance.

Table 3.1-1 Number of Images versus Encoded Level

Images Encoded to	LOS	BEZ	Contractor
9-digit Direct	402	68	119
9-digit HA	202	71	106
9-digit S	219	68	123
5 digits	96	165	97
3 digits	8	83	40
Error Add-on	0	14	51
Error 5-digit	0	95	119
Unresolved	0	363	272
Total	927	927	927

This distribution of number of images versus encoded level can also be used in conjunction with a cost function to establish a costing model. This model can then be used to compare the LOS, BEZ, and ERIM systems. The cost model for the multiline optical character reader was given per 1,000 mailpieces as:

9-digit direct	4.50
9-digit H-coded	13.02
9-digit S-coded	21.53
5-digit coded	36.63
3-digit coded	45.26
Error Add-on	62.98
Error 5-digit	88.14
Unresolved	51.79

Combining this function with the distribution of images given in Table 3.1-1 yields the costs shown below in Table 3.1-2.

Table 3.1-2 Mail Processing Cost

Images Encoded to	LOS	BEZ	Contractor
9-digit Direct	1809.00	306.00	535.50
9-digit HA	2630.04	924.42	1380.12
9-digit S	4715.07	1464.04	2648.19
5 digits	3516.48	6043.95	3553.11
3 digits	362.08	3756.58	1810.40
Error Add-on	0.00	881.72	3211.98
Error 5-digit	0.00	8373.30	10488.66
Unresolved	0.00	18799.77	14086.88
Total	13032.67	40549.78	37714.84

The third tabular measure of performance is shown in Table 3.1-3. This table breaks down the system analysis based on categories of MLOCR performance. The rows represent a level of sortation achieved by the MLOCR on a set of images. The columns then represent our performance within this category of sortation.

Table 3.1-3 Contractor's Performance versus BEZ Results

BEZ Results	Number of Images	% Correct	% Reject	% Err. Add-on	% Err. 5-Digit
9-Digits	207	84.06	0.00	9.66	6.28
5 Digits	165	87.88	0.00	7.88	4.24
3 Digits	83	86.75	1.20	7.23	4.82
Rejects	363	17.63	74.38	3.31	4.68
Error Add-on	14	35.71	0.00	57.14	7.14
Error 5-digit	95	12.63	0.00	6.32	81.05
Total	927	50.92	29.23	7.01	12.84

Table 3.1-4 shows a confusion matrix where the rows represent ERIM's performance as broken down into categories of sortation and the columns represent the LOS results as broken down by the same categories of sortation. The two columns and rows before the "Total" column/row are error categories where "Add" implies that the base 5-digit ZIP was correct but there was an error in the 4-digit add-on, and "5-d" represents an error in the 5-digit base ZIP Code. In general, positive entries in the upper triangle of this matrix represent improvements over the LOS results and lower triangular numbers indicate lesser performance than the LOS results. Since the LOS results represent an upper bound, this matrix is almost completely lower triangular.

Similar to Table 3.1-4, Table 3.1-5 also shows a confusion matrix where the rows represent levels of sortation achieved by ERIM's system. The two tables differ in the fact that the columns in this new table represent levels of sortation achieved by the MLOCR as opposed to the Electrocom LOS results. As contrasted with the previous matrix, this matrix is much denser in its upper triangle than lower, which indicates a significant performance improvement than just the MLOCR results alone.

3.2 LOS Discrepancies

As was mentioned previously, LOS discrepancies need to be accounted for to accurately measure the true performance of the end-to-end system. For our initial run, 29 LOS discrepancies were identified in the 72 errors. When these results are included , the system level results without using the BEZ increased to 319 (34.34%) correct, 43 (4.63%) error, and 565 (60.95%) rejects. The tables presented in section 3.1 were regenerated from the post-processed results and are given in Tables 3.2-1 to 3.2-5.

Table 3.1-4 Distribution of Images vs. Level of Sort --- Electrocom's LOS Results

	9-Di	9-HA	9-H	9-S	5	3	Rej	Add	5-d	Total
9-Di	119	0	0	0	0	0	0	0	0	119
9-HA	1	42	0	0	0	0	0	0	0	43
9-H	14	8	41	0	0	0	0	0	0	63
9-S	8	0	0	115	0	0	0	0	0	123
5	41	17	11	12	16	0	0	0	0	97
3	20	8	3	5	4	0	0	0	0	40
Rej	93	24	23	61	64	7	0	0	0	272
Add	27	2	8	10	4	0	0	0	0	51
5-d	79	13	2	16	8	1	0	0	0	119
Total	402	114	88	219	96	8	0	0	0	927

Table 3.1-5 Distribution of Images vs. Level of Sort --- MLOCR Results

	9-Di	9-HA	9-H	9-S	5	3	Rej	Add	5-d	Total
9-Di	55	0	5	1	11	13	22	5	7	119
9-HA	1	24	0	0	10	3	3	0	2	43
9-H	0	0	36	1	7	7	11	0	1	63
9-S	0	0	0	64	20	11	26	0	2	123
5	0	0	0	1	96	0	0	0	0	97
3	0	0	0	0	1	39	0	0	0	40
Rej	0	0	0	0	0	0	272	0	0	272
Add	0	0	6	0	13	6	12	8	6	51
5-d	12	0	0	1	7	4	17	1	77	119
Total	68	24	47	68	165	83	363	14	95	927

Table 3.2-1 Number of Images vs. Encoded Level

Images Encoded to	LOS	BEZ	Contractor
9-digit Direct	402	68	142
9-digit HA	202	71	108
9-digit S	219	68	126
5 digits	96	165	97
3 digits	8	83	40
Error Add-on	0	14	36
Error 5-digit	0	95	106
Unresolved	0	363	272
Total	927	927	927

Table 3.2-2 Mail Processing Cost

Images Encoded to	LOS	BEZ	Contractor
9-digit Direct	1809.00	306.00	639.00
9-digit HA	2630.04	924.42	1406.16
9-digit S	4715.07	1464.04	2712.78
5 digits	3516.48	6043.95	3553.11
3 digits	362.08	3756.58	1810.40
Error Add-on	0.00	881.72	2267.28
Error 5-digit	0.00	8373.30	9342.84
Unresolved	0.00	18799.77	14086.88
Total	13032.67	40549.78	35818.45

Table 3.2-3 Contractor's Performance BEZ Results

BEZ Results	Number of Images	% Correct	% Reject	% Err. Add-on	% Err. 5-Digit
9-Digits	207	92.27	0.00	7.25	0.48
5 Digits	165	90.30	0.00	5.45	4.24
3 Digits	83	87.95	1.20	6.02	4.82
Rejects	363	19.01	74.38	2.20	4.41
Error Add-on	14	35.71	0.00	57.14	7.14
Error 5-digit	95	13.68	0.00	5.26	81.05
Total	927	53.94	29.23	5.39	11.43

Table 3.2-4 Distribution of Images vs. Level of Sort --- LOS Results

	9-Di	9-HA	9-H	9-S	5	3	Rej	Add	5-d	Total
9-Di	136	0	6	0	0	0	0	0	0	142
9-HA	1	42	0	0	0	0	0	0	0	43
9-H	15	9	41	0	0	0	0	0	0	65
9-S	8	0	0	116	2	0	0	0	0	126
5	41	17	11	12	16	0	0	0	0	97
3	20	8	3	5	4	0	0	0	0	40
Rej	93	24	23	61	64	7	0	0	0	272
Add	22	1	2	9	2	0	0	0	0	36
5-d	66	13	2	16	8	1	0	0	0	106
Total	402	114	88	219	96	8	0	0	0	927

Table 3.2-5 Distribution of Images vs. Level of Sort --- BEZ Results

	9-Di	9-HA	9-H	9-S	5	3	Rej	Add	5-d	Total
9-Di	67	0	10	1	13	13	25	5	8	142
9-HA	1	24	0	0	10	3	3	0	2	43
9-H	0	0	36	1	9	7	11	0	1	65
9-S	0	0	0	64	20	12	28	0	2	126
5	0	0	0	1	96	0	0	0	0	97
3	0	0	0	0	1	39	0	0	0	40
Rej	0	0	0	0	0	0	272	0	0	272
Add	0	0	1	0	9	5	8	8	5	36
5-d	0	0	0	1	7	4	16	1	77	106
Total	68	24	47	68	165	83	363	14	95	927

3.3 Control Level Refinements

Much of the month prior to Phase II testing was spent completing new and improved system level components and integrating them into the Phase II system. Since this integration took longer than anticipated, little time was left before the Phase II test to perform a complete system checkout. Because of this, the Phase II test results uncovered minor control level refinements that were needed to fully realize the improvements made to the individual components. This section describes the affect of these changes on system performance. The Phase II test images were processed one final time with all of the modifications included. The results from this test without using the BEZ can be summarized as follows: 407 (43.81%) of the images received a correct 9-digit ZIP Code, 39 (4.20%) of the images received an incorrect 9-digit ZIP Code, and 483 (51.99%) images were rejected by our system.

The five tabular methods of calculating system level performance for this final run are included below in Tables 3.3-1 to 3.3-5.

Table 3.3-1 Number of Images vs. Encoded Level

Images Encoded to	LOS	BEZ	Contractor
9-digit Direct	402	68	157
9-digit HA	202	71	129
9-digit S	219	68	153
5 digits	96	165	80
3 digits	8	83	39
Error Add-on	0	14	39
Error 5-digit	0	95	98
Unresolved	0	363	232
Total	927	927	927

Table 3.3-2 Mail Processing Cost

Images Encoded to	LOS	BEZ	Contractor
9-digit Direct	1809.00	306.00	706.50
9-digit HA	2630.04	924.42	1679.58
9-digit S	4715.07	1464.04	3294.09
5 digits	3516.48	6043.95	2930.40
3 digits	362.08	3756.58	1765.14
Error Add-on	0.00	881.72	2456.22
Error 5-digit	0.00	8373.30	8637.72
Unresolved	0.00	18799.77	12015.28
Total	13032.67	40549.78	33484.93

Table 3.3-3 Contractor's Performance BEZ Results

BEZ Results	Number of Images	% Correct	% Reject	% Err. Add-on	% Err. 5-Digit
9-Digits	207	91.79	0.00	7.25	0.97
5 Digits	165	88.48	0.61	6.06	4.85
3 Digits	83	92.77	1.20	4.82	1.20
Rejects	363	28.93	63.64	3.03	4.41
Error Add-on	14	35.71	0.00	64.29	0.00
Error 5-digit	95	21.05	0.00	4.21	74.74
Total	927	58.58	25.13	5.72	10.57

Table 3.3-4 Distribution of Images vs. Level of Sort --- Electrocom's LOS Results

	9-Di	9-HA	9-H	9-S	5	3	Rej	Add	5-d	Total
9-Di	148	2	6	1	0	0	0	0	0	157
9-HA	1	58	0	0	0	0	0	0	0	59
9-H	16	9	44	0	1	0	0	0	0	70
9-S	5	0	0	140	8	0	0	0	0	153
5	33	12	10	13	12	0	0	0	0	80
3	21	5	4	3	6	0	0	0	0	39
Rej	91	17	18	41	58	7	0	0	0	232
Add	27	1	2	6	3	0	0	0	0	39
5-d	60	10	4	15	8	1	0	0	0	98
Total	402	114	88	219	96	8	0	0	0	927

Table 3.3-5 Distribution of Images vs. Level of Sort --- BEZ Results

	9-Di	9-HA	9-H	9-S	5	3	Rej	Add	5-d	Total
9-Di	66	0	10	4	20	14	26	4	13	157
9-HA	1	24	0	0	13	8	12	0	1	59
9-H	0	0	35	1	11	5	16	1	1	70
9-S	0	0	0	62	23	14	49	0	5	153
5	0	0	0	1	79	0	0	0	0	80
3	0	0	0	0	1	37	1	0	0	39
Rej	0	0	0	0	0	0	232	0	0	232
Add	0	0	1	0	10	4	11	9	4	39
5-d	1	0	1	0	8	1	16	0	71	98
Total	68	24	47	68	165	83	363	14	95	927

3.4 Independent Test Set Results

In order to test the robustness of the system modifications discussed in section 3.3, an independent test set of address block images was gathered. 959 previously unprocessed images were gathered from the training 1 dataset and completely processed by the end-to-end system. The results without BEZ can be summarized as follows: 403 (42.02%) were correctly processed, 37 (3.86%) were incorrectly processed, and 519 (54.12%) images were rejected. These are numbers that are quite comparable to the final results on the Phase II test images and represent a substantial improvement over the Phase I prototype system. The five tables for this testing procedure are given below.

Table 3.4-1 Number of Images vs. Encoded Level

Images Encoded to	LOS	BEZ	Contractor
9-digit Direct	280	32	103
9-digit HA	221	87	150
9-digit S	272	64	177
5 digits	126	200	120
3 digits	23	89	36
Error Add-on	0	15	28
Error 5-digit	0	63	69
Unresolved	37	409	276
Total	959	959	959

Table 3.4-2 Mail Processing Cost

Images Encoded to	LOS	BEZ	Contractor
9-digit Direct	1260.00	144.00	463.50
9-digit HA	2877.42	1132.74	1953.00
9-digit S	5856.16	1377.92	3810.81
5 digits	4615.38	7326.00	4395.60
3 digits	1040.98	4028.14	1629.36
Error Add-on	0.00	944.70	1763.44
Error 5-digit	0.00	5552.82	6081.66
Unresolved	1916.23	21182.11	14294.04
Total	17566.17	41688.43	34391.41

Table 3.4-3 Contractor's Performance BEZ Results

BEZ Results	Number of Images	% Correct	% Reject	% Err. Add-on	% Err. 5-Digit
9-Digits	183	93.99	0.00	4.37	1.64
5 Digits	200	95.50	0.00	1.50	3.00
3 Digits	89	91.01	0.00	5.62	3.37
Rejects	409	26.16	66.75	1.47	5.62
Error Add-on	15	60.00	0.00	40.00	0.00
Error 5-digit	63	38.10	0.00	3.17	58.73
Total	959	60.90	28.47	3.13	7.51

Table 3.4-4 Distribution of Images vs. Level of Sort --- Electrocom's LOS Results

	9-Di	9-HA	9-H	9-S	5	3	Rej	Add	5-d	Total
9-Di	91	2	2	1	6	1	0	0	0	103
9-HA	2	74	0	0	0	0	0	0	0	76
9-H	19	3	47	3	1	1	0	0	0	74
9-S	7	0	3	162	4	1	0	0	0	177
5	23	11	12	30	44	0	0	0	0	120
3	18	3	4	5	5	1	0	0	0	36
Rej	81	18	28	46	54	16	33	0	0	276
Add	17	3	1	5	2	0	0	0	0	28
5-d	22	6	4	20	10	3	4	0	0	69
Total	280	120	101	272	126	23	37	0	0	959

Table 3.4-5 Distribution of Images vs. Level of Sort --- BEZ Results

	9-Di	9-HA	9-H	9-S	5	3	Rej	Add	5-d	Total
9-Di	27	0	4	5	24	14	18	3	8	103
9-HA	0	40	4	2	15	4	8	2	1	76
9-H	2	0	35	1	10	2	18	2	4	74
9-S	0	0	1	53	21	27	62	2	11	177
5	0	0	0	0	120	0	0	0	0	120
3	0	0	0	0	1	34	1	0	0	36
Rej	0	0	0	0	0	0	276	0	0	276
Add	1	0	2	3	3	5	6	6	2	28
5-d	2	1	0	0	6	3	20	0	37	69
Total	32	41	46	64	200	89	409	15	63	959

4 Plans for Phase III

The two major objectives for phase III are: 1) to improve system performance through refinement of the current system components, and 2) to incorporate the use of MLOCR character recognition data into the contextual analysis system. Eight major areas of work have been identified as defining a path towards achieving these two goals. The Appendix details these major tasks in a Gantt chart format. The major areas are also broken down into subtasks in this appendix. This section of the report gives a brief description of the work required for each major task and its associated subtasks.

4.1 MLOCR Data Integration

The use of MLOCR recognition data will be one of the major challenges of phase III. One part of the problem is matching the recognition data to the images elements (lines, words, and characters) segmented by the contextual analysis system. This should eventually be made simpler by the availability of coordinate information from the MLOCR itself, but for now it remains a problem.

A more fundamental problem is to develop strategies for using the character recognition information at various stages within the current system.

One obvious place for MLOCR data is in numeral reading. The digit recognition module could either rely tot illy on the number reading from the MLOCR, or it could augment its own scheme with the ML( )CR data.

Another use for MLOCR data is in word verification. It seems reasonable to assume that some increased accuracy in word verification, and the associated confidence assignment problem, can be made through the use of MLOCR character recognition data.

A third possible use of alphabetic character recognition in is the generation of specific word queries into the address database. This could be especially effective if some words are read with very high confidence.

Finally, the investigation of number vs. num-number field discrimination using the MLOCR recognition channels should be investigated. This type of information could assist greatly in the parsing of an input address block image.

4.2 Segmentation and Image Processing

The operation of the character segmentation module must be improved. The word verification module only works if the word is segmented correctly. The approach to be used is for the character segmentation module to produce several hypothesis segmentations. Then, word verification can use the segmentation which corresponds to the length of the string being verified.

The use of multiple character segmentation hypotheses also has impact on the database query module, in that word lengths are one of the query items. Multiple word length hypotheses lead to alternate query sets.

The word segmentation module might also be changed to generate multiple hypotheses in some circumstances. These alternate segmentations could be parsed separately, and would lead to more alternate parses.

Image quality measurements, and associated methods for image repair might be added to the contextual analysis system. These methods would help the processing of highly degraded images.

A conservative punctuation detection and removal algorithm currently exists in the Phase II system. The knowledge of the location of possible or definite punctuation is confined, however, to the word segmentation subsystem. This knowledge could assist greatly in parsing the address block. It could also assist in the generation of queries where often an incorrect word length query is made due to punctuation being incorrectly counted as an extra character.

4.3 Numeral Reading

The Phase II Context System still employs the use of the ECA supplied image truth to ascertain the values of the numeric fields in the image (i.e. street number, secondary number, etc.). This is clearly not an implementable solution. While the integration of MLOCR character recognition offers a partial solution to this problem, a more robust solution can be found through the combination of the MLOCR information with techniques developed at ERIM.

A first attempt will involve the training of a backpropagation neural network with inputs derived from the cavity features developed for the OCR research project. A decision strategy will have to be developed that attempts to find a satisfactory operating point for the various usages of the number recognition system.

One of these possible usages could be to assist in the determination of numeric versus non-numeric fields in the address block. It was previously mentioned that such a capability could greatly assist in parsing. This capability could also help in reducing system errors by directing number recognition to numeric fields only.

A significant number of mailpieces have their bottom line cut off, either by an envelope window, or by the edge of the image sensing field. It might be useful to build specialized recognition of this instance, and call specialized processes into play. For example, we might explore the development of a separate number reading module which reads numbers only looking at their top half. This could be useful in identifying the ZIP code, or at least reducing the possible set of ZIP codes.

4.4 Word Verification

Early in Phase III, the character segmentation subsystem will be modified to generate multiple ranked hypotheses of the correct character boundaries of an input word. The word verification subsystem will need to be modified so as to take advantage of these multiple hypotheses.

The word verification module also needs better confidence measures. The difference in probability between the top two hypotheses should be considered, as well as the difference between the top choice, and the maximum probability achievable. Also, image quality measurements might be used to augment or bias confidence assignment processes.

New feature extraction techniques will also be investigated. These techniques will take into account the baseline of a candidate word in order to better focus the system to areas of interest on the character. This task will also assist in the classification of an input word's case; Upper, Lower, Mixed, or Unknown.

4.5 Address Block Parsing

The parsing system currently in place makes use of "key" word recognition (such as suffix names, suite, box, and other address words) to assign confidence to the various parse hypotheses. The a priori statistics for address block formats are not used for this purpose. The addition of format statistics could help in parse ranking. This might involve the development of more sophisticated ranking methods. (Use fuzzy logic, Dempster/Shaffer approach to ranking alternatives?)

The address block parser must be refined to include more address block formats. This can be done by augmenting the current address block models. Main areas of interest in this task are the modelling of Rural Route's and more relaxed methods of covering currently modelled mailpieces. Possible punctuation location will also be incorporated at this level.

As was mentioned previously, the current parser relies heavily on specific lexicons to identify key words in the address block image. These lexicons will be expanded in Phase III to include more potential words. One possible problem with using the lexicon approach is that some words appear in more than one lexicon (e.g. box is a post office box word and a building secondary word). A strategy will be developed to deal with these multi-role words.

The word segmentation subsystem will also be modified in Phase III to generate multiple word segmentation hypotheses. Parsing will have also have to be modified accordingly to take advantage of this opportunity.

4.6 Queries and Database

Several current queries exist that use word length as their key into the database. The generation of multiple word length hypotheses due to multiple character segmentation dictates the modification of these queries to allow for multiple values.

New queries are also an avenue that needs investigation. A field in the address block that has been ignored until now has been the street number. Two possible queries related to this field have been suggested: the number of digits in the street number, and the first two digits in the street number.

Perhaps the biggest effect of all on new queries will be due to the availability of MLOCR character information. Querying on a specific character at a precise location in a word will often be a very strong focussing piece of information. The frequency and robustness of this information will have to be investigated.

The use of word replacement strategies (sometimes called a thesaurus) could help the system in cases where the form of a word does not match the form found in the ZIP+4 directory. This will also be an area of development in Phase III.

The integration of the National Carrier Walk Sequence data into the databases and processing strategy has possibilities for improving reader performance. In particular, the job of reading street numbers could become one of verification rather than recognition.

4.7 System Decision Module

The overall control strategy currently in place only codes the mailpiece to 9 digits or rejects it. In the case of a reject, the result of the MLOCR machine is used. This is a rather simple control strategy and an obvious area for improvement for Phase III. First, a control strategy will need to examine both the ERIM answer and the MLOCR answer when both are available and find a compromise solution to resolving any possible conflict. The ERIM system also needs a capability to encode mailpieces to 5 or 3 digits when the MLOCR is in error or has rejected the piece. These are issues that could greatly decrease the dollar cost associated with system performance

The current system decision confidence is based solely on word verification confidence of the primary name. The development of a address level verification module, which assigns a confidence to the overall interpretation of the address block could help to decrease system errors.

4.8 System Testing and Evaluation

We have internally broken up Phase III into two development periods and two system testing and evaluation periods. Each development period will consist of five weeks and will be followed by a three week system evaluation period. The evaluation period is directed at identifying areas where the system needs improvement and redirecting research.

The official Phase III system testing period has been tentatively scheduled for sometime in November. We have set aside the last two months cf Phase III for System testing and reporting tasks although work on the system may continue during this time.

5 Bibliography

[1] Davis, R. H. and Lyall, J. Recognition of handwritten characters - a review. *Image and Vision Computing*. 4(4):208-218, November 1986.

[2] Gronmeyer, L. K., Ruffin, B. W., Lybanon, M. A., Neely, P. L., and Pierce, S. E. An Overview of Optical Character Recognition (OCR) Technology and Techniques. *Technical Report 217, Naval Ocean Research and Development Activity*, NSTL Station, Mississippi 39529, June 1978.

[3] Harmon, L. D. Automatic recognition of print and script. *Proceedings IEEE*, 1165-1175, October 1972.

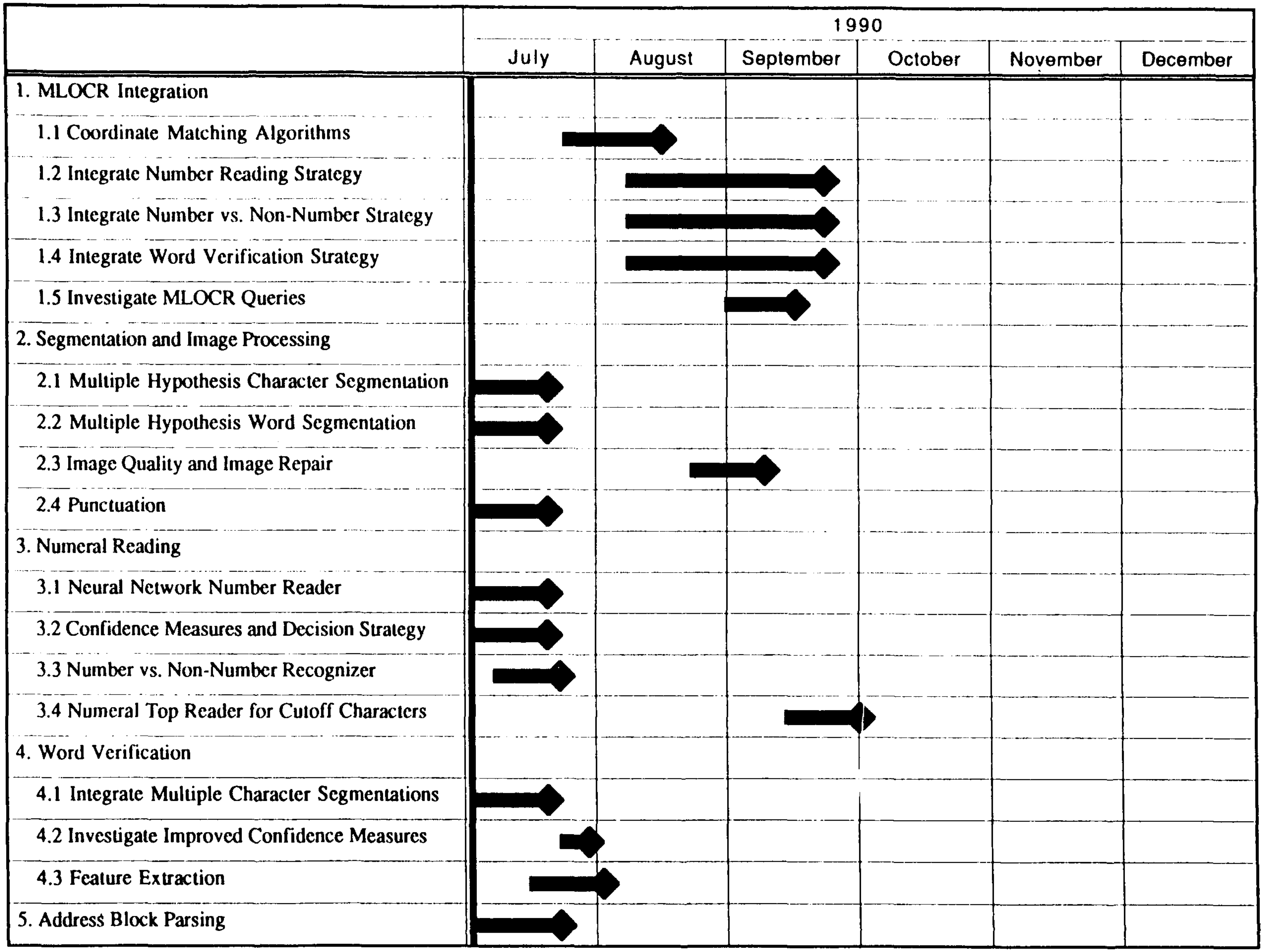
[4] Mantas, J. An overview of character recognition methodologies. *Pattern Recognition*, 19(6), 425-430, 1986.

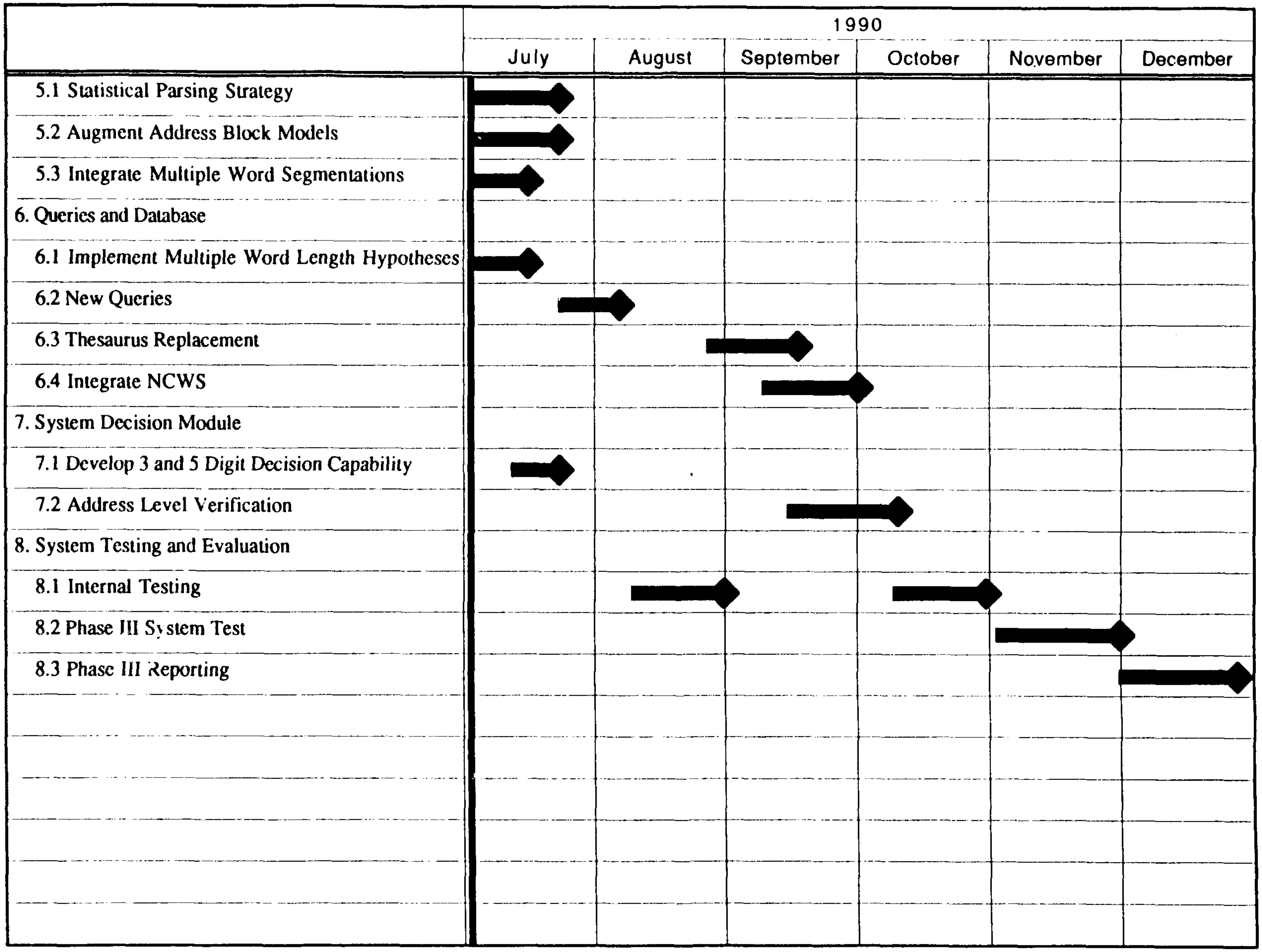
[5] Suen, C. Y., Berthod, M., and Mori, S. Automatic recognition of handprinted characters - the state of the art. *Proceedings of the IEEE*, 68(4):469-487, April 1980.

[6] Stentiford, F.M. W., Automatic feature design for optical character recognition using evolutionary search procedures. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 3, pp 349-355, 1985.

[7] Gillies, A. "Automatic generation of morphological template features", *Proceedings of the SPIE*, vol. 1350, *Image Algebra and Morphological Image Processing*, 1990.

Appendix
Phase III Schedule


1990
July
August
September
October
November
December
1. MLOCR Integration
1.1 Coordinate Matching Algorithms
1.2 Integrate Number Reading Strategy
1.3 Integrate Number vs. Non-Number Strategy
1.4 Integrate Word Verification Strategy
1.5 Investigate MLOCR Queries
2. Segmentation and Image Processing
2.1 Multiple Hypothesis Character Segmentation
2.2 Multiple Hypothesis Word Segmentation
2.3 Image Quality and Image Repair
2.4 Punctuation
3. Numeral Reading
3.1 Neural Network Number Reader
3.2 Confidence Measures and Decision Strategy
3.3 Number vs. Non-Number Recognizer
3.4 Numeral Top Reader for Cutoff Characters
4. Word Verification
4.1 Integrate Multiple Character Segmentations
4.2 Investigate Improved Confidence Measures
4.3 Feature Extraction
5. Address Block Parsing


1990
July
August
September
October
November
December
5.1 Statistical Parsing Strategy
5.2 Augment Address Block Models
5.3 Integrate Multiple Word Segmentations
6. Queries and Database
6.1 Implement Multiple Word Length Hypotheses
6.2 New Queries
6.3 Thesaurus Replacement
6.4 Integrate NCWS
7. System Decision Module
7.1 Develop 3 and 5 Digit Decision Capability
7.2 Address Level Verification
8. System Testing and Evaluation
8.1 Internal Testing
8.2 Phase III System Test
8.3 Phase III Reporting